PLANEACION Y EVALUACION DEL APRENDIZAJE A TRAVES DEL COMPUTADOR

JORGE ENRIQUE ECHEVERRY

HERNAN VILLAMARIN

Trabajo de Grado presentado como requisito parcial para optar al título de Ingeniero Industrial.

Director: DARIO ESPINOZA

CORPORACION UNIVERSITARIA AUTONOMA DE OCCIDENTE

DIVISION DE INGENIERIAS

PROGRAMA DE INGENIERIA INDUSTRIAL

CALI, 1987

AGRADECIMIENTOS

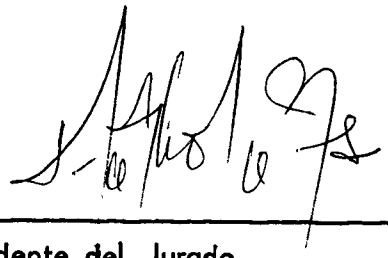
Los Autores expresan sus agradecimientos :

A CARLOS BELTRAN, Ingeniero

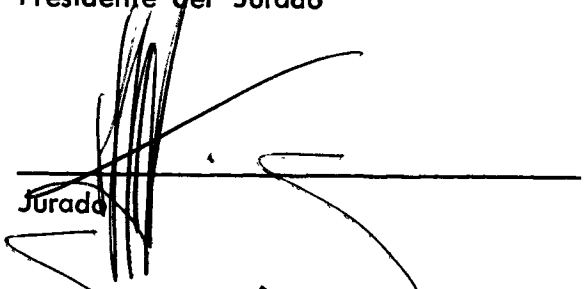
A DARIO ESPINOSA, Ingeniero, Jefe del Departamento de Sistemas de la Corporación Universitaria Autónoma de Occidente.

A todas aquellas personas que en una u otra forma colaboraron en la realización del presente trabajo.

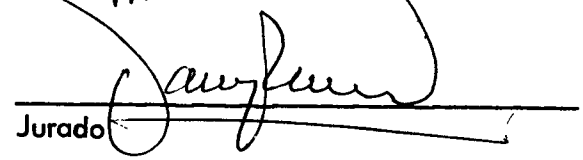
Aprobado por el Comité de trabajo de Grado en cumplimiento de los requisitos exigidos por la Corporación Universitaria Autónoma de Occidente para otorgar el título de Ingeniero Industrial.



Presidente del Jurado



Jurado



Jurado

Cali, Noviembre de 1987

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

En general el contenido del proyecto consta de 5 capítulos a través de los cuales se va desarrollando el mismo, al tiempo que está sujeto a la ejecución práctica del trabajo, es decir, la obtención de los programas (paquete) del Sistema. En el primer capítulo titulado el Computador en la Educación - Generalidades, se tratan aspectos históricos y se clasifican los usos que se dá al computador en la enseñanza buscando con ello clarificar nuestra situación. En un segundo capítulo iniciamos entonces una descripción amplia del sistema que se ha de generar, y de las condiciones y características del sistema , bajo el título Descripción General del Sistema, en el que se busca definir el ambiente de operación , equipos , entorno, software y Hardware.

Iniciamos entonces el tercero y más importante de los capítulos del trabajo El Diseño , en el que mediante los Diagramas de Flujos de Datos (DFD) se busca definir la estructura lógica del sistema de acuerdo a todas las características definidas para su posterior programación y funcionamiento óptimo. Finalizado esto se inicia entonces la programación en el cuarto capítulo , en la cual se ha codificado en un lenguaje de Alto Nivel (TurboPascal) todas las rutinas o acciones del sistema para interpretación por parte del computador.

Verificada las bondades del sistema y su funcionamiento óptimo mediante repetidas pruebas, se inicia el quinto y último capítulo en el que se ofrece un Manual del Usuario con el fin de hacer más fácil la utilización del sistema .

En el se describe la forma de operación y se dan las pautas para el uso y programación de los cursos.

INTRODUCCION

Es indudable que asistimos como espectadores y simples usuarios a los avances tecnológicos y teorías innovadoras acerca de los computadores, y los últimos logros con ellos en variedades de aplicaciones.

En la Educación, hemos visto como es ampliamente utilizado en nuestro medio, en aplicaciones administrativas, que tienen que ver con exámenes de admisión, calificaciones, registro de estudiantes, asignación de cursos, profesores, horarios, etc.; también como herramientas intelectual en el estudio de materias que por su naturaleza, requieren de su uso, y, como objeto de estudio en diferentes materias de la Ingenierías, especialmente de la Ingeniería de Sistemas, y de la Ingeniería Electrónica, pero no al nivel creativo y productivo de los países altamente industrializados que los producen, sino a un nivel de uso y mantenimiento.

Pero existe una cuarta forma de utilización del computador en la enseñanza y es como medio para enseñar las diferentes áreas del conocimiento, es decir, como máquina de enseñanza. "Máquina de enseñanza, es un conjunto específico de funciones o transformaciones usado para producir cambios de comportamiento sistemáticos en un estudiante, cuyas respuestas al material presentado determinan la subsiguiente operación del mecanismo,

es decir, retroalimentan el sistema .

No solo en nuestro medio Colombiano es subutilizado el computador en esta actividad, sino a nivel mundial, tanto que en estudios realizados hacia 1984 y 1985 en los Estados Unidos y en Gran Bretaña sobre el uso del computador en la enseñanza primaria y secundaria, arrojaron resultados desalentadores.

La explicación a esto, se encuentra tal vez en la complejidad de los sistemas y Software existentes, puesto que se requiere de programadores especializados en los lenguajes de autoría y tutoría desponibles para la creación de sistemas de este tipo que puedan ser utilizados para "Enseñar y Evaluar". Esto, dificulta el uso del computador en esta actividad, puesto que son pocas las Entidades docentes que se pueden dar el lujo de tener profesionales especializados en estos lenguajes, destinados a crear tutores para materias específicas, de acuerdo a los programas académicos de sus cursos.

Nosotros proponemos en este proyecto, un pequeño aporte a la solución de este problema, con un sistema de autoría, que no está basado en uno de estos lenguajes especializados autoriales, sino en un lenguaje de comandos sencillos, que permite a los profesores corrientes, después de conocer un poco el sistema y el manejo sencillo del Editor de Textos del sistema operacional DOS, la programación de su curso, sin necesidad de ser un experto programador.

Buscamos con este proyecto la elaboración de un prototipo autorial (Sistemade Instrucción Asistida por Computador. CAI), que se use en la Corporación Universitaria Autónoma de Occidente, o en cualquier centro docente, como un elemento que facilite el proceso del aprendizaje y que mida los resultados que se obtengan en dicho proceso.

Al entrar a sistematizar el proceso de Enseñanza - Aprendizaje, debe considerarse todos aquellos conceptos que se encuentran asociados estrechamente a él, como serían los objetivos específicos , resultados esperados del proyecto (Sistema) , los criterios básicos de evaluación, las pautas estratégicas a seguir en la conformación de los cursos (textos), la evaluación y actualización de los instrumentos de medición de el progreso de los alumnos, y, otras condiciones específicas de un sistema pedagógico .

Es indispensable tener en cuenta también la influencia del factor humano en este proyecto de sistematización del proceso, es decir, los usuarios del sistema que son dos básicamente . Un Instructor o Profesor, que es el usuario No. 1 y un estudiante que es el usuario No. 2.

En la actualidad el sistema educativo normal, cuenta con ciertas características que han sido objeto de modificaciones en diferentes ocasiones, con éxito en algunos de los casos, pero regresando casi siempre sobre sus características generales de Acción que son las conocidas hoy por hoy.

De estas características, se desprenden como las más sobresalientes por ejemplo el hecho de que es el profesor quien lleva la parte activa del

sistema, y quien lleva la iniciativa en el proceso de enseñanza - aprendizaje, además de la responsabilidad y el control de las herramientas disponibles para llevar a cabo el proceso. Su función es de informador, puesto que se limita a transmitir sus conocimientos en clases magistrales o grupos muy grandes, y se encuentra con inconvenientes como la falta de tiempo, diferencias en los niveles de cada estudiante de su grupo con los demás, debidas a su preparación individual previa al curso, tamaño del grupo, a su posición en el salón de clases, etc.

El alumno se limita, en actitud pasiva, a aceptar lo que le transmite su profesor. No cuenta con una atención especial o dedicación individual por parte del profesor, sino que los 30, 40 ó 50 alumnos de un grupo, reciben igual tratamiento como si fuesen un grupo homogéneo. No hay tiempo por parte del profesor para dedicarse a los alumnos con dificultades especiales en el aprendizaje.

En el sistema actual, la labor de enseñanza, resulta monótoma y en cierta forma repetitiva y con pocas posibilidades de innovación, puesto que el tiempo del profesor, lo limita a sus clases magistrales y a la revisión de evaluaciones, que es considerada por muchos de ellos, larga y tediosa. Esto redunda en desventajas para las dos partes humanas del proceso, ya que dificulta al profesor la tarea de mantener actualizados los registros del progreso de los alumnos.

Otro de los grandes problemas a través del desarrollo educativo, ha sido la forma estática, como en muchos casos, se lleva a cabo el progreso

no servía a nuestros propósitos.

sobre los contenidos programáticos de los cursos y la forma repetitiva como los educadores transmiten año tras año los conocimientos , debido en gran parte al poco tiempo del que dispone el educador para la parte creativa.

En el sistema propuesto por nosotros, en el que en cierta forma se sistematiza el proceso, se busca que el alumno , se enfrente a los temas sin la presencia constantemente del profesor, para acrecentar su responsabilidad en el proceso de aprendizaje. Esta no-presencia contínua del Profesor, le beneficia a él también, puesto que le permite dedicar gran parte de su capacidad a los estudiantes en desventajas y al control del nivel del curso.

El profesor continúa siendo el responsable del curso, en cuanto a instrucción pero ahora comparte, o mejor, entrega al alumno la responsabilidad del aprendizaje para que este se autodiscipline.

Como es obvio , al estudiante se le entrega la posibilidad de llevar la iniciativa , por lo tanto se espera de él responsabilidad, desarrollo de su creatividad y de su capacidad de pensamiento crítico. Para este fin , se dá al instructor en el sistema, la posibilidad de uso de unos mecanismos informadores diversos y completos, y diferentes posibilidades de evaluación.

Este proyecto fué desarrollado iniciando con una recopilación de fuentes bibliográficas a cerca de la Educación y el uso de los computadores en ella. Posteriormente ésta se clasificó y se desechó todo el material que no servía a nuestros propósitos.

Además de esto fueron fuentes de consulta aproximadamente 30 profesores de planteles de Educación primaria, media y universitaria, con los que se realizaron charlas informales con ciertos puntos patrones como temas. Posteriormente procedimos a resumir la historia y uso del computador en la enseñanza y a continuación se desarrolló el proyecto en el orden en que se presentan los capítulos, pues este orden obedece al desarrollo cronológico de las labores desarrolladas.

En apartes internos se profundiza sobre la metodología llevada a cabo para el diseño y ejecución del sistema que es objetivo fundamental de este proyecto.

Adjunto a la tesis se hace entrega de un Diskette que contiene los programas que se desarrollaron para obtener el paquete PLEAC1. Estos programas son Registro y Presenta. La clave de acceso consiste en digitar 4 veces la tecla ENTER.

1. EL COMPUTADOR EN LA EDUCACION

1.1 GENERALIDADES

En este capítulo se refleja el desarrollo histórico de los Computadores en la educación hasta llegar a los sistemas tutoriales y autoriales que se desprenden de este desarrollo como los más regentes y apropiados de los sistemas CAI, y un paralelo entre ellos.

Los Sistemas CAI (Computer Asistent Instruction) son aquellos sistemas en los que se lleva a cabo el proceso de instrucción entre el hombre y la computadora, en el cual el hombre aprende y la máquina es controlada por los programas de enseñanza almacenados ensu memoria, diseñados para informar, guiar, controlar y examinar al estudiante hasta que este alcance un determinado nivel de aprendizaje en la materia en cuestión.

Por lo tanto, todos los sistemas tratados en adelante desarrollan sistemas CAI.

Se hablará de la Introducción del Microcomputador o P.C. y el avance que significó su llegada para los sistemas CAI, y por último de algunas consideraciones de la tecnología educativa.

1.2 USO PRECEDENTE DEL COMPUTADOR EN LA ENSEÑANZA

En esta Sección nos vamos a referir a los aspectos netamente académicos en que puede intervenir un Computador en la enseñanza.

Nos remontamos al siglo XIX cuando en 1866 Halcyon Skinner desarrolló y patentó una "Máquina de Ortografía"que servía como herramienta al profesor ayudando a resolver problemas de lógica presentados en forma de símbolos.

Ya en este siglo en el año de 1926 Siderney L. Pressey, Sicólogo - Pedagogo, considerado el pionero en la enseñanza automatizada , introdujo una máquina que "Imparte y callfica exámenes y enseña", basada en el principio fundamental (o técnica básica) de aprendizaje "Estímulo Respuesta - Refuerzo" y que hoy fundamenta la mayoría de los modelos actuales de CAI.

A pesar de que este aparato no llegaba siquiera a tener las cualidades que su creador le asignaba, introdujo sobre firmes bases unos principios y conceptos muy útiles para el desarrollo posterior de el tema que aquí tratamos.

Nos adentramos entonces en un período de algo menos de 40 años sólo hasta el año de 1950 - 54 B.F. Skinner dirigió nuevamente investigaciones sobre la Educación invididualizada. Habían aparecido numerosas herramientas y medios de los que debían valerse para lograr progresos , como la TV., el cine, fonográfo y grabadoras y así ocurrió en efecto

con las ayudas visuales.

Para Skinner la máquina podía enseñar no sólo mediante un examen, sino independientemente.

Se dió un gran paso. El material presentado por las máquinas ya no solo era simple ejercicio - práctica. Ahora introducían también el contenido teórico y ejemplarizado de la materia a evaluar posteriormente. El interés se centraba en la transmisión de la información y no en la evaluación.

Al material programado se le concedían ya características tutoriales.

1.3 DESARROLLO DE LENGUAJES Y SISTEMAS

En el año de 1958 la IBM. fabricó una máquina que combinaba el computador digital IBM-650 yuna máquina de escribir eléctrica cuyo fin era la enseñanza de la aritmética binaria (RATH ET AL 1959).

Este indicaba, validando dígito a dígito, si la respuesta estaba errada, disminuyendo la ocurrencia de errores, y seleccionanbo los nuevos problemas, basándose en la cantidad de errores cometidos por el usuario. Así un usuario que prácticamente estaba libre de errores, podía seguir con la instrucción.

En 1965 la Universidad de Illinois, junto con la IBM, desarrolló un lenguaje de autoría al que le llamó PLATO (Programad Logic for Automatic

Teaching Operation) . El cual combinaba el uso del computador con cintas grabadas y el video.

En 1968, Jhon Starkweather en asocio con la Universidad de California, introdujo el lenguaje PILOT, especializado para el diseño de lecciones de educación programada asistida por computador (Starkweathen Programed Inquiry, Learning or Teaching).

PILOT, permite desarrollar contenidos en un ambiente programable, capaz de hacer decisiones basándose en las respuestas del estudiante .

Actualmente PLATO, es uno de los dos sistemas más grandes de enseñanza por computador, que contiene varios niveles de tipo tutorial , en los cuales se interactúa directamente (Estudiante - computador), para impartirle el material, ayudarle y proveerle ejercicios (Marcos , Juán , 1969)

PILOT, está conformado por un autor y un tutor. El tutor es un interpretador de lenguaje Pilot , y el autor está constituído por cuatro editores que soportan la elaboración de los textos, conjuntos, de caracteres, gráficas en color y composiciones musicales o efectos de sonido .

El lenguaje tiene recursos para hacer preguntas al usuario, analizar sus respuestas, presentar gráficas y producir secuencias musicales, basándose en los recursos desarrollados en los 4 editores.

La versión original del PILOT que funciona sobre un sistema de tiempo com-

partido , SDC 940, se inspiró a su vez en un programa anterior, el "Computest", frecuentemente utilizado en los High School en los EE. UU.; del que después se desarrolló la versión para microcomputadores.

En el campo de los computadores personales o micros, se ha evolucionado con sistemas y lenguajes utilizados como herramientas para la educación entre los cuales figuran los Sistemas TICCIT, el Lenguaje LOGO y el Lenguaje KAREL.

El Sistema TICCIT - ("Time shared Interactiva Computer Controller Information Television "), es una forma de aproximación a la problemática de la preparación de material de enseñanza en el computador. Aquí ya no existe un lenguaje de autoría si no más bien un sistema de autoría que le proporciona al instructor una serie de marcos sobre los cuales trabajar y sobre los cuales se encuentra definida una estructura lógica fija. El instructor interactúa con el sistema, el cual le va preguntando qué es lo que debe pasar de acuerdo a la estructura que se encuentra definiendo.

En este modelo la idea del trabajo en grupo, entre personas vinculadas a la instrucción programada y al sistema, es más clara, incluso recomendada en el desarrollo de los materiales . Aún así el material producido mediante este modelo no ha sido suficiente para dar un concepto definido acerca del modelo .

LOGO es un lenguaje orientado a permitir una gran flexibilidad en los manejos de listas y gráficas , detalle éste último que genera un gran atrac-

tivo en los usuarios de corta edad y por lo tanto una gran facilidad en el proceso del aprendizaje . La deficiencia de este lenguaje radica en que al ser utilizado para otros campos de la enseñanza , como por ejemplo series de ejercicio en nociones de geometría y aritmética, nos encontramos con que requiere de un amplio conocimiento del lenguaje por parte del profesor , lo que no es muy usual.

Uno de los campos en que podría ser bien utilizado y se presta muy bien para este propósito es como medio de ejercicio en la enseñanza de elementos de programación y computadores.

KAREL es un sistema que posiciona un robot en un mundo imaginario y permite al usuario desarrollar en él, programas en un lenguaje sencillo y primitivo, que tienen el fin específico de llevar al robot karel a conseguir el cumplimiento de el objetivo específico planteado en el ejercicio.

1.4 EL MICROCOMPUTADOR

Hasta los últimos sistemas tratados en la sección anterior, la tecnología ofrecida como último avance los "sistemas de tiempo compartido", que innegablemente lograron con su empuje una mayor expansión y difusión del computador. Pero estos equipos no tenían características llimitadas y no lograron vencer los inconvenientes más relevantes propios de ellos como lo costoso que resultaban. El tipo de instalaciones que requerían, equipos grandes y procesamiento central de datos, lentitud en el tiempo de respuesta, caída del sistema, etc.

En este momento apareció el computador personal que dejó atrás todas estas barreras, permitió en el campo de los sistemas CAI un mayor desarrollo y les abrió grandes perspectivas en la investigación.

Los Sistemas CAI que se habían limitado a sistemas de Tiempo Compartido, ahora contaban con una nueva herramienta para la presentación y desarrollo del material de enseñanza a los alumnos (Franklin , Stephen) .

El micro se presenta no solo como la solución a ciertas limitaciones del Sistema de TC, sino también como reemplazo total, ya que tiene la posibilidad de instalarse como terminal de un sistema de TC, o lo que es mejor una instalación de microcomputador con disco duro, impresora y pantalla de alta resolución, que mejora indiscutiblemente el sistema de TC para los fines que persiguen los sistemas CAI (Farber Fair, Uhlig).

Algunas ventajas que presenta el P.C. frente al sistema de T.C., de importancia para los sistemas CAI, son :

La persona que hace uso del equipo tiene acceso a todos los recursos del mismo, en cualquier momento y por el tiempo que desee.

El PC. trabaja por si solo y cualquier falla en el sistema afecta solo al usuario.

Los costos de adquisición son bajos y aún tienden a bajar.

La distribución y utilización del PC es mucho más inmediata y opera en cualquier lugar .

Los problemas de comunicación de estos con uno de cualquier otro tipo se están solucionando rápidamente.

La inclusión de facilidades gráficas en el Hardware y el Software está altamente generalizada .

1.5 SISTEMAS AUTORES

Se plantea ya no el desarrollo de lenguajes autoriales, sino de sistemas de autoría, que le proporcionan al usuario (Instructor) una serie de marcos sobre los cuales trabaja y sobre los que se encuentra definida una estructura lógica fija; el instructor también como el alumno interactúa con el sistema que le pregunta qué es lo que debe ocurrir de acuerdo con la estructura que en ese momento define .

Veamos como podríamos definir la Autoría; " Son todas aquellas actividades que tengan como fin desarrollar materiales de enseñanza por computador, incluída su planeación y archivo ".

El objetivo central de los S.A. es simplicar la interacción entre el computador y el usuario y provee al instructor una herramienta simple y acorde con sus necesidades para la generación de contenidos programáticos dinámicos , es decir , sujetos a cambios con el tiempo y las necesidades .

Una de las consecuencias que se espera de la elaboración de un sistema de autoría es la no utilización de la programación de computadores para la generación o creación de los contenidos permitiendo entonces su utilización por profesores no profesionales en el área de sistemas.

Básicamente el sistema autor debe contener dos grandes conjuntos de función:

Uno que permite al instructor crear y generar los materiales de enseñanza, el orden del contenido programático y evaluaciones para ser presentados en pantalla;

Y otro que desarrollo tal presentación , es decir, que imparte las lecciones en base a la representación del contenido almacenada en los archivos creados por el sistema .

Para lograr el funcionamiento de un sistema de este tipo, lo ideal es, no que una persona realice todo el trabajo, sino que sea un grupo el que lo lleve a cabo. Este grupo estaría compuesto al menos por un instructor o profesor , un programador de instrucción y el programador de computadores, que debe tener un amplio conocimiento del lenguaje de autoría. Este modelo de acción ha sido utilizado en el proyecto PLATO con la desventaja de que una sola persona se encarga de las funciones de definición , planeación , codificación e implementación del material en el computador . Esto genera a su vez uno de los grandes factores negativos del PLATO: La persona demora muchos años en llegar a dominar el lenguaje con el

consiguiente detrimento para los materiales de enseñanza produc

1.6 SISTEMAS TUTORES

Son sistemas de enseñanza escritos para satisfacer propósitos muy específicos. A diferencia de los sistemas autoriales no brindan flexibilidad dinámica para modificar sus contenidos ni para cambiarlos, y menos para elaborar uno nuevo, pues para lograrlo se debe crear otro sistema tutor diferente de nuevo.

Los S.T. están orientados al diálogo con el estudiante y presupone para su ejecución una alta dosis de sistemas de autoría, desarrollados normalmente por equipos de instructores y programadores.

En general, un tutor completo consta sólo de el conjunto de recursos que ejecuta la presentación de contenidos en uno o más medios de salida del equipo. La diferencia de este punto con el Autor es que aquí es mucho más completo (especializado), que el del Sistema Autor. Cabe aclarar que los contenidos y su representación quedan elaborados desde la etapa de programación, es decir, son parte de ella.

TABLA 1. Autores y Tutores - Paralelo

Sistema Tutor	Sistema Autor
Debido al carácter estático de la información que transmite y su conocimiento previo por parte del programador se aprovechan mejor los recursos de la máquina para ofrecer mejores recursos de aprendizaje.	No cuentan con esta ventaja de aprovechamiento óptimo de recursos, pero ofrecen una buena gama de ellos suficiente para mucho tipo de recursos o temas.
Requiere capacitación profesional del Instructor en Sistemas y programación.	No requiere conocimientos previos de programación por parte del instructor.
Su mayor desventaja es su primera ventaja: El conocimiento que transmiten es estático (Muy difícil de modificar).	Carácter dinámico del material de enseñanza: Lo puede modificar el instructor, con gran facilidad, sin necesidad de cambiar el programa solo el contenido.
Altos costos en modificaciones del material pedagógico tiempo de programadores y de máquina.	Bajos costos, no requiere mayor tiempo de máquina y programadores.
Requiere equipo de trabajo mínimo de 3 personas para su creación.	Requiere de 1 persona para su creación, ya que el instructor se considera usuario.
Gran capacidad computacional que permite al Instructor manipular respuestas y le dan flexibilidad para alterar la secuencia de instrucción de acuerdo a las respuestas del estudiante .	Poca o casi nula capacidad computacional, como resultado del objetivo de eliminación de la programa como una necesidad para el desarrollo de los contenidos.

2. DESCRIPCION GENERAL Y ANALISIS DEL SISTEMA

En el presente capítulo pretendemos introducir ya la creación del sistema prototipo de nuestro proyecto: PLEAC I .

El nombre genérico hace referencia al título de esta Tesis "Planeación y Evaluación del aprendizaje a través del Computador" . Es un sistema prototipo del campo autorial; y el I introduce este sistema como base para que sea perfeccionado en proyectos posteriores, por ejemplo introduciendo en el sistema la parte correspondiente a gráficas que ha sido una de las limitantes de nuestro trabajo.

Trataremos en apartes de este capítulo los objetivos de PLEAC I, una breve descripción del sistema, su capacidad, los requerimientos de operación (Hardware) y algunas especificaciones y recomendaciones dirigidas a quienes eventualmente desean modificar o mejorar este proyecto.

2.1 ALCANCE DEL SISTEMA

Bajo las dos limitantes yá tratadas anteriormente, en cuanto a graficación que justifica todo un proyecto de tesis, y la de baja capacidad computacional, propia de los sistemas y /o lenguajes autores, definiremos el obje-

tivo, descripción y capacidad del sistema.

2.1.1 Descripción del Sistema

PLEAC I está compuesto básicamente por dos programas: REGISTRO y PRESENTA, que utilizan una serie de rutinas y permiten su utilización por dos usuarios (Instructores y alumnos), de la siguiente manera: El instructor tiene acceso al programa Registro, para apertura de los diferentes archivos del sistema, para registrar estudiantes y para crear los textos de las lecciones de su curso y sus correspondientes evaluaciones, así mismo para consultar el progreso de sus alumnos (notas), sus respuestas a las evaluaciones y calificarlas en casos que lo requieran.

El estudiante tiene acceso al programa PRESENTA, el cual le muestra por pantalla el contenido del curso página por página y diferentes tipos de evaluaciones orientadas a reforzar su aprendizaje y a evaluarlo.

Este sistema está sujeto a ser utilizado para numerosas áreas de la enseñanza debido a que no hay limitación de este tipo puesto que el texto, cualquiera que sea se almacena en formato estandar de archivos de texto ASC-II y se desarrollan mediante el uso simple de un editor de texto o procesador de palabra.

2.1.2 Objetivo

El objetivo de estos programas es permitir al usuario instructor sin necesidad

de poseer conocimientos en sistemas, la generación de un curso en determinada área del conocimiento (materia), con un contenido teórico textual. Ejemplo, Ejercicios y Evaluaciones, para que sus alumnos (usuarios también del sistema), reciban la presentación de estos textos, aprendan la teoría mediante ejemplos prácticos y ejerciten y refuercen el conocimiento adquirido a través de los ejercicios. Además permitir al profesor medir y consultar en cualquier momento al progreso de sus alumnos.

Todo esto orientado a que el profesor pueda brindar personalmente a cada alumno el tiempo que gana al no tener que dedicar todo su tiempo a la clase magistral.

2.1.3 Capacidad del Sistema

2.1.3.1 Capacidad Operacional

La capacidad operacional de PLEAC I está basada en tres tipos de operaciones (segmentos de texto).

Un texto de exposición que se edita por medio del DOS y se expone en la pantalla para su lectura, un texto de operación computacional ó de operaciones matemáticas básicas que se puedan almacenar en variables, y un texto para prueba o evaluación que presenta un ejercicio al que se le adjudica posteriormente o en el momento de responder una nota basada en su porcentaje de validez.

2.1.3.2 Capacidad de Almacenamiento

La capacidad de los archivos de PLEAC I, debido al tratamiento del turbo Pascal es tan grande como registros permitan uá sea el Disquette si se trabaja en un microcomputador, o el Disco Duro si lo tiene y en éste está instalado PLEAC I.

El curso a programar puede contener un máximo de 75 lecciones, cada una de ellas con la capacidad de uno de los archivos descritos anteriormente y a su vez cada una de estas lecciones pueden contener ya sea lecciones o sublecciones y así mismo estas últimas pueden contener a otras y así sucesivamente. Estos registros asemejan el tipo de enumeración utilizada en este trabajo.

2.1.4 Utilización y Alternativas

La concepción de uso inicial bajo la que diseñamos este sistema, es su utilización en la enseñanza en planteles de cualquier nivel (primarios - secundarios - universitarios - de postgrado) para la formación de los alumnos en las áreas de estudio a que se de lugar. Pero puede también ser utilizado de otras formas, como por ejemplo en validación de materias a nivel universitario, en presentación de pruebas de admisión, como material de consulta (al tener programados cursos específicos) en las bibliotecas, y cuantos usos lógicos se les pueda ocurrir en el campo académico teniendo en cuenta las limitaciones del sistema.

2.2 REQUERIMIENTOS PARA OPERACION

En esta sección trataremos de definir claramente los equipos, el medio, el entorno y condiciones bajo las que debe operar el sistema, y en los cuales ha sido creado.

2.2.1 HARDWARE

PLEAC I requiere para su operación, equipos con un mínimo de 256 K de memoria RAM y unidad de Diskette, sin ser necesaria una impresora. Se recomiendan equipos PC IBM ó compatibles con este (en el que fué con cebido y programado). Uno de los factores relevantes para la buena operación del sistema, es el conocimiento del equipo utilizadi, por quien introduzca y genere contenidos.

Debemos recordar que el sistema no maneja gráficas y que una solución a esto sería la utilización de material adjunto suministrado por el instructor.

2.2.2 Software

PLEAC I utiliza en su programación un editor de texto del sistema operacional DOS, lo que inhibe la utilización de procesadores de palabra, ya que para editar una lección el instructor solo debe utilizar la opción EDITAR del menú que se le presenta y el sistema lo ubica directamente en el editor al que puede introducir textos y evaluaciones que seran guardados en formatos ASC II estandar.

2.2.3 Aspectos básicos de operación

Es no solo recomendable sino necesario que los textos a generar, sean preparados previamente por el instructor de acuerdo a las lecciones que se van a presentar y su orden.

PLEAC I se utilizará entonces como el Software para impartir las lecciones preparadas por el profesor ó instructor.

Otro aspecto importante para que el texto generado sea admitido por PLEAC I, es el hecho de que se ajuste a las normas que aparecen en el manual de usuario (capítulo 5.).

Recalcamos lo dicho en la sección 2.2.1 de este capítulo acerca de que es necesario y recomendable el conocimiento del equipo (teclado y funciones principalmente) por parte de quien introduce los textos y evaluaciones para la operación del sistema.

Por último una recomendación ya no tanto de operación, sino académica, pero relevante, es la revisión periódica del rendimiento académico de los alumnos por medio del sistema, para hacer los ajustes necesarios en los textos y reforzar el aprendizaje de aquellos que lo requieran, además de identificar a cuales de los alumnos se les debe dedicar más tiempo.

2.3 LENGUAJE Y METODOLOGIA DE PROGRAMACION

Aquí se describe la forma en que fue implementado el programa, las pruebas y el lenguaje utilizado para la programación. Esto está dirigido a programadores y/o estudiantes de la CUAO., que efectuen revisiones del Software descrito en este proyecto, o que inicien otro proyecto con la ventana que dejamos abierta para PLEAC II: Adición de manejo de gráficas al actual sistema.

En la programación ufé utilizado el Lenguaje Turbo Pascal versión 3.0. " Comentando las muchas mejoras del Turbo Pascal un crítico dijo que este no es realmente Pascal, sino un nuevo lenguaje como el Pascal. Ciertamente es verdad que el Turbo Pascal permite, realmente facilita, actividades nunca permitidas por el profesor NICKLAUS WIRTH, creador del Pascal, por otra parte,es posible escribir programas en el entorno Turbo Pascal de acuerdo con el Pascal standar ISO ".

El turbo Pascal combina un editor , un compilador y un depurador para crear un entorno de desarrollo Software muy productivo.

La versión utilizada es completa de acuerdo al manual de referencia del Turbo.

La metodología seguida para el desarrollo de PLEAC I fué la siguiente:

Inicialmente se identificaron las características relevantes que deseabamos

fueran parte del sistema , como el interpretador de comandos o Paiser de comandos, graficación , textos, tipos de evaluación o ejercicios, disminución de la desventaja de sistemas autores en cuanto a NULA capacidad computacional, generando la opción de cómputo con operaciones básicas , formas de acceso al sistema de los diferentes usuarios: Profesores - alumnos , y en general las características actuales de sistemas de lenguajes autoriales.

La parte de graficación fué desechada desde un principio , ya que como hemos dicho anteriormente es bastante compleja y justificada "Todo un proyecto".

Una vez definidas estas características entramos en la parte de diseño en la que se definieron a nivel general y de detalle 1 los módulos necesarios para la operación. Para este fin se utilizaron los diagramas de flujos de datos DFD, de la tecnología de la Empresa MOBIL OIL Co, de los que conocíamos su facilidad de uso , claridad, versatilidad y facilidad de transporte al lenguaje definitivo.

Por último se inició la parte de programación en la que se fué desarrollando cada uno de los módulos básicos, agregando a los programas en forma ordenada las caracteristicas requeridas, se digitaba, se compilaba y entonces se hacían las pruebas hasta obtener plena satisfacción del funcionamien to del programa con las últimas adiciones. En síntesis las labores correspon dientes a la escritura de algoritmos detallados, digitación, compilación y pruebas se efectuaron poco a poco al compas del desarrollo del programa.

3. DISEÑO

Como veíamos en la sección 2.3 del capítulo 2, la metodología utilizada para el diseño del sistema PLEAC I es la de diagrama de flujos de datos DPD'S de la Mobil OIL Co., en su parte de herramientas y técnicas para análisis y el diseño de sistemas de información.

Pues bien, los diagramas de flujos de datos DFD'S se definen como una representación en RED de un sistema en términos de sus componentes y las conexiones entre estos. Se trata de un método gráfico que facilita el desarrollo descendente y se utilizan como una herramienta para aumentar y formalizar el conocimiento.

No son un objeto en sí.

Constan de cuatro componentes básicos : (ver figura 1.)

Estos componentes tienen algunas características especiales como porejemplo:

Los almacenamientos de datos son compartidos al menos entre dos procesos. En ellos se debe identificar las claves de acceso o modificación si las hubiere.

Se debe bautizar ó identificar todas las entidades, procesos, flujos y almacenamiento de datos.

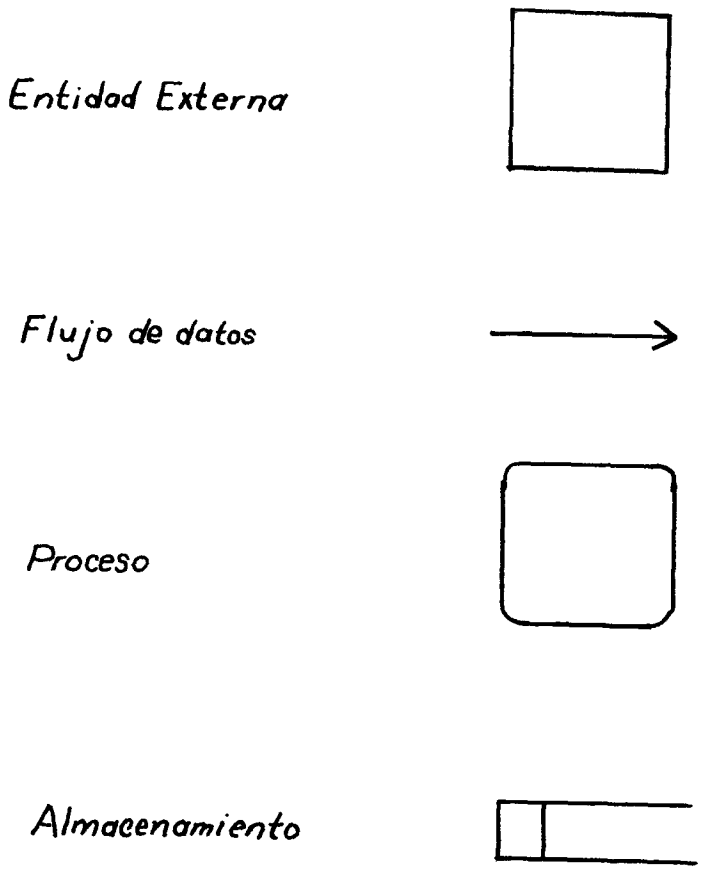


FIGURA 1. Simbología de los DFD'S

Los Procesos:

Deben tener un número de referencia único.

Deben ser definidos con verbos activos. Ejemplo: No procesar sino Procesa.

No Editar , sino Edita.

Los subprocesos deben tener igual identificación.

Deben tener una breve descripción.

Es opcional que se identifique en ellos el departamento o programa que efectúa el proceso.

Facilitan agrupamientos y desgloses.

Los flujos de Datos:

Deben tener una dirección explícita y única.

No deben existir entre entidades externas y almacenamiento.

Pueden existir duplicaciones en las entidades externas y los almacenamientos de datos para facilitar el seguimiento.

Estas normas se resumen en la figura 2.

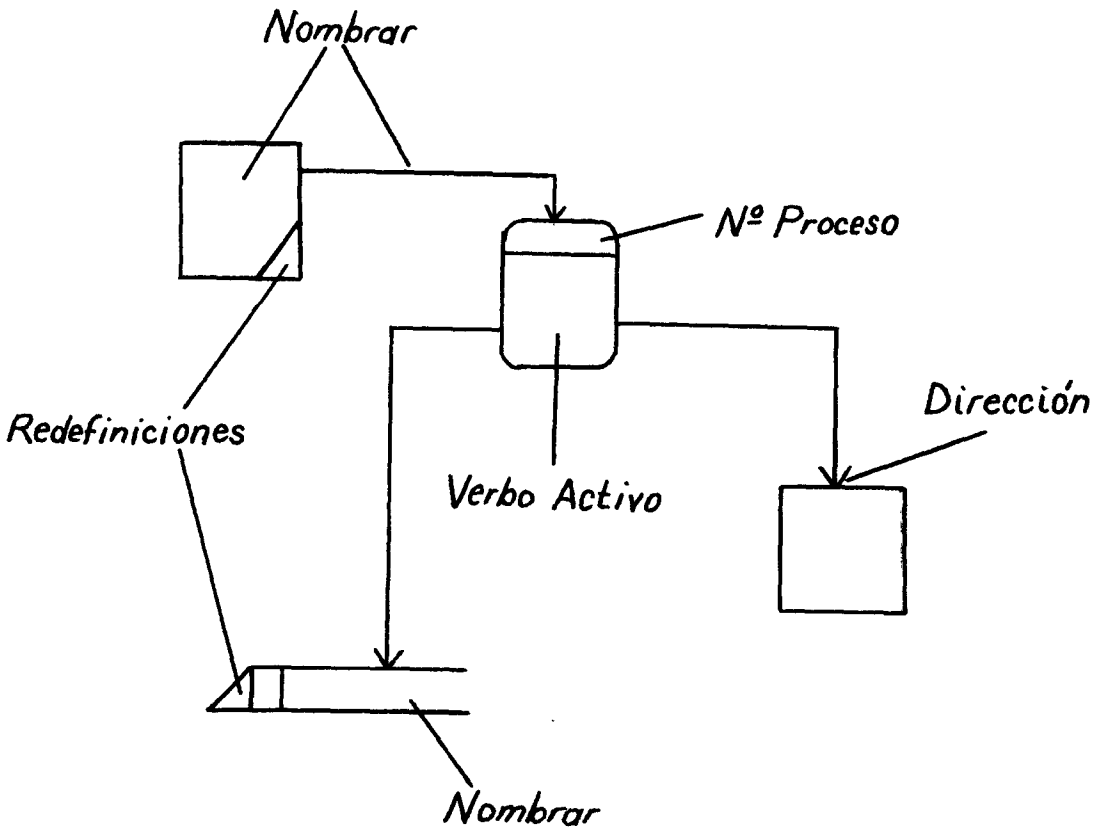


FIGURA 2. Reglas y Regulaciones de los DFD.S

Los niveles de los DFD'S son 4:

Un nivel de contexto en el que se define el alcance del sistema.

Contiene un solo proceso para todo el sistema y no involucra almacenamiento de datos.

Un nivel Cero, que es una visión general del sistema. En el se determinan las funciones del sistema y se identifican los flujos de datos, procesos para cada función y almacenamiento de datos.

Un tercer nivel uno, en el que están ya definidos los procesos, almacenamientos y entidades externas, describiendo un mayor nivel de detalle. Generalmente de 2 a 8 procesos por cada función de nivel cero.

Y un nivel dos, que puede incluirse en el gráfico del nivel uno, en el que hace una descripción del sistema por subniveles y subprocesos, en el se identifican los subprocesos y se desarrolla el diagrama para cada subproceso ó subsistema. Pueden existir desde 2 hasta 20 subprocesos por proceso de nivel uno.

Al comenzar a descender de un nivel a otro se presenta la explosión de los procesos uno a uno (Explote). Esta facilita la representación de subprocesos en niveles más bajos, aquí encontramos otras tres reglas.

Todos los flujos que entran y salen del nivel superior deben existir en los niveles inferiores y viceversa.

Los flujos "internos" a un proceso no aparecen en los niveles superiores.

Los subprocesos de un proceso deben identificarse con el mismo número , más una notación decimal (9 - 9.1, 9.2, 9.2.1, 9.2.2, etc.).

En este capítulo de Diseño se mostrarán y aplicarán los diferentes niveles de los DFD'S del sistema y luego las estructuras de información , es decir unos cuadros que conforman los MENUS PRINCIPALES, las rutinas empleadas en PLEAC I, y la secuencia en que van siendo llamadas (Diagramas de árbol).

3.1 DFD - NIVEL DE CONTEXTO DE PLEAC I

En el nivel de contexto de PLEAC I, nos encontramos con un único proceso que aprende, enseña y evalúa , con lo que nos referimos a que recibe la edición, ejercicios y evaluaciones del curso (Aprende), presenta los textos por pantalla al alumno y los ejercicios prácticos (evalúa).

Este proceso recibe y envía flujos de datos desde y hacia 3 entidades externas que son : El Instructor, el Estudiante y el Administrador, que dado el caso puede ser el mismo instructor ó profesor.

Los flujos de datos y conformación de este nivel de contexto podemos apreciarlo en la Figura 3.

3.2 DFD NIVEL CERO DE PLEAC I

En este nivel Cero encontramos ya una visión general de PLEAC I. En el

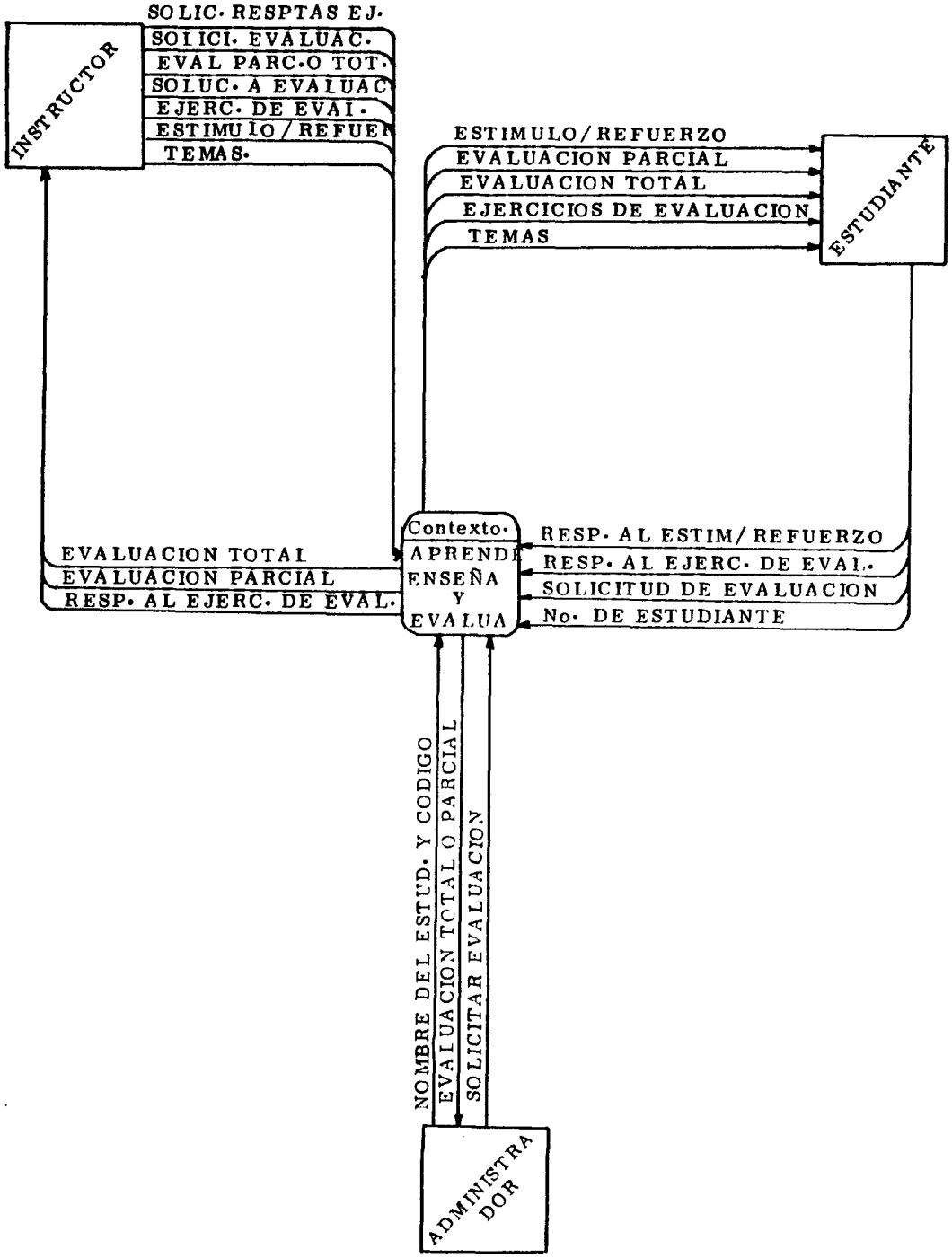


FIGURA 3. Nivel de Contexto de PLEAC1

están las funciones del sistema representadas en los siguientes procesos:

Un editor de textos (EditaTemas - 1.0) que edita los temas y los lleva al archivo del curso.

Una autorización de acceso del estudiante al sistema (Autoriza - 2.0) que verifica el hecho de que el estudiante esté registrado en el curso.

Un informador de progreso de los estudiantes (Informa Progreso - 3.0) que puede ser consultado por el profesor y forma información del archivo.

Un presentador de respuestas (presenta Respuestas - 3.0) que consulta las respuestas dadas por los estudiantes a las evaluaciones en el archivo de respuestas dadas por los estudiantes a las evaluaciones en el archivo de respuestas a ejercicios y las presenta al profesor.

Un expositor (Expone - 5.0) que presenta los temas por pantalla tomándolos del archivo del curso.

Y un proceso de registro de estudiantes al sistema ó al curso (Registra estudiante - 6.0) que inscribe al estudiante en el curso dentro del archivo de desempeño.

La constitución de estos procesos, los archivos de los que hablamos , y los flujos de datos entre ellos y las entidades externas las podemos apreciar en la figura 4.

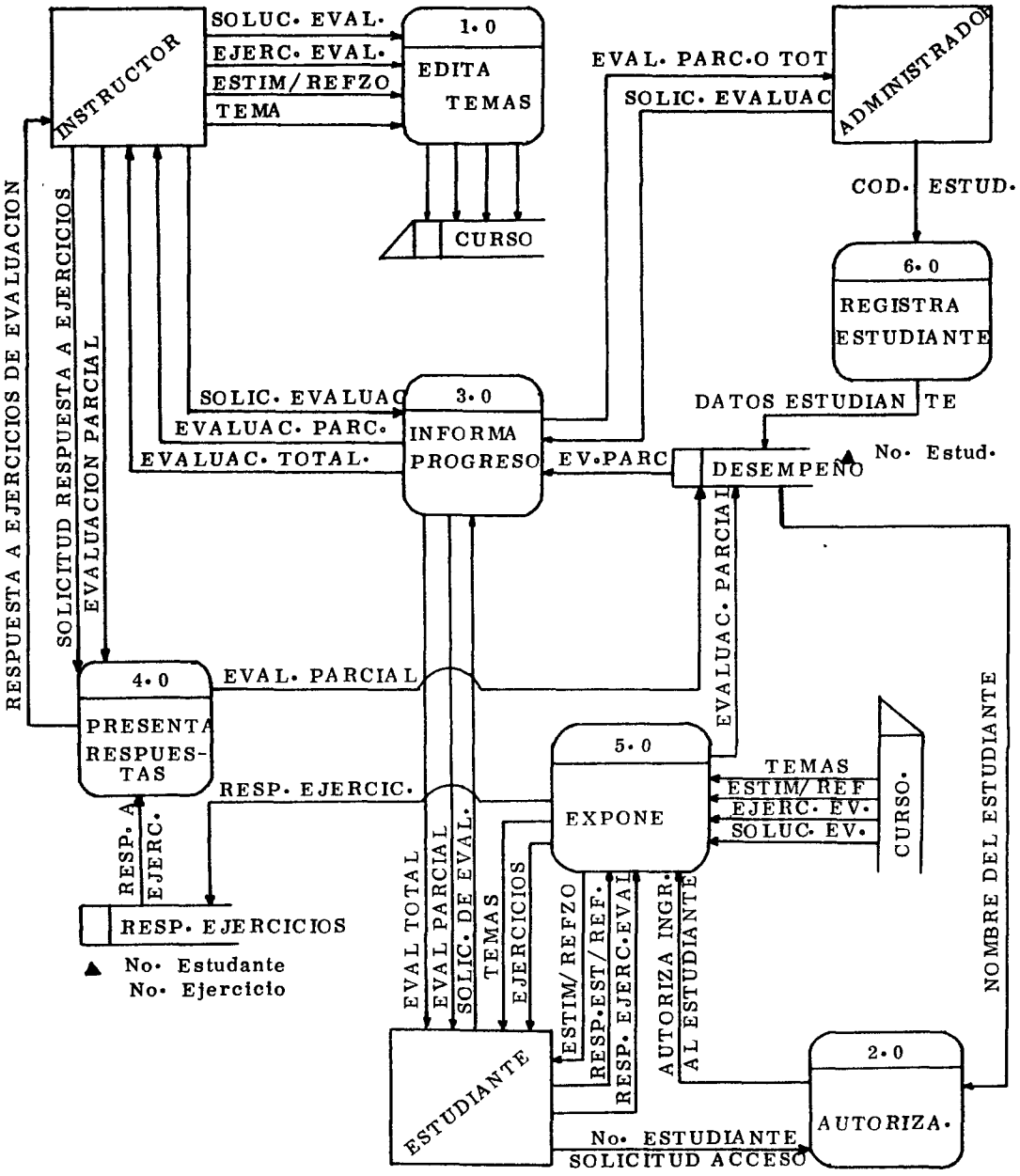


FIGURA 4. Nivel Cero de PLEAC1

3.3 DFD'S NIVEL UNO DE PLEAC I

Entramos yá al "EXPLOTE" *de cada uno de los procesos del nivel Cero.*

3.3.1 Explote 1.0 (Editatema)

Cuando veíamos el capítulo 2 en la sección 2.2.2, haciamos referencia al Software utilizado por el sistema y dijimos que PLEAC I utiliza el editor de textos del sistema operacional DOS, por lo que no es necesario el explote del proceso "EDITA TEMA".

3.3.2 Explote 2.0 (*Autoriza*)

Según se puede apreciar en el nivel Cero (figura 4), este proceso solo tiene flujo de datos desde y hacia la entidad "estudiante" y lo suple con la consulta al archivo de desempeño. Esta única función no amerita otro nivel para este proceso por lo que no se hace necesario tampoco el explote en nivel uno.

3.3.3 Explote 3.0 (Informa progreso)

El nivel Uno interno del proceso 3.0 está compuesto por cuatro procesos que confirman la acción del informe progreso (consulta) que son : Un validador de solicitudes del Instructor en cuanto al progreso de los estudiantes que verifica la existencia del estudiante en el archivo de desempeño (VALIDA SOLICITUD 3.1). Un proceso que tramita esa solicitud de

evaluación parcial (Tramita solicitud evaluación parcial 3.3); Uno que tramita y/o calcula solicitudes de evaluación total (Calcula Evaluación total 3.2); Y uno que obtiene las evaluaciones parciales del archivo de desempeño (Obtiene Evaluación Parcial 3.4), yá sea para entregarlas a 3.3 como parciales ó a 3.2 para que calcule la evaluación total. Este proceso en su nivel 1 se puede apreciar claramente en la figura 5.

3.3.4 Explote 4.0 (Presenta Respuestas)

En este proceso de presentar respuestas , se ejecutan dos acciones importantes (Procesos). Al ser solicitado por el instructor presenta las respuestas que han dado todos los estudiantes a las preguntas de respuesta abierta de determinada lección (obtiene respuestas 4.1) y dá al instructor la opción de calificar inmediatamente esa respuesta (Entra evaluación Parcial Manual 4.2) ó de continuar con las siguientes respuestas, para calificar posteriormente.

El proceso se puede apreciar claramente en la figura 6.

3.3.5 Explote 5.0 (Expone)

El proceso Expone, es el que presenta por pantalla el texto del curso de acuerdo a como se encuentra este en el archivo del curso. Esto lo hace a solicitud del estudiante por medio del proceso PRESENTA TRAMAS 5.1, pero necesita de un traductor, que convierta el lenguaje que introduce el estudiante en algo asimilable por el contenido del texto, ó interpretador

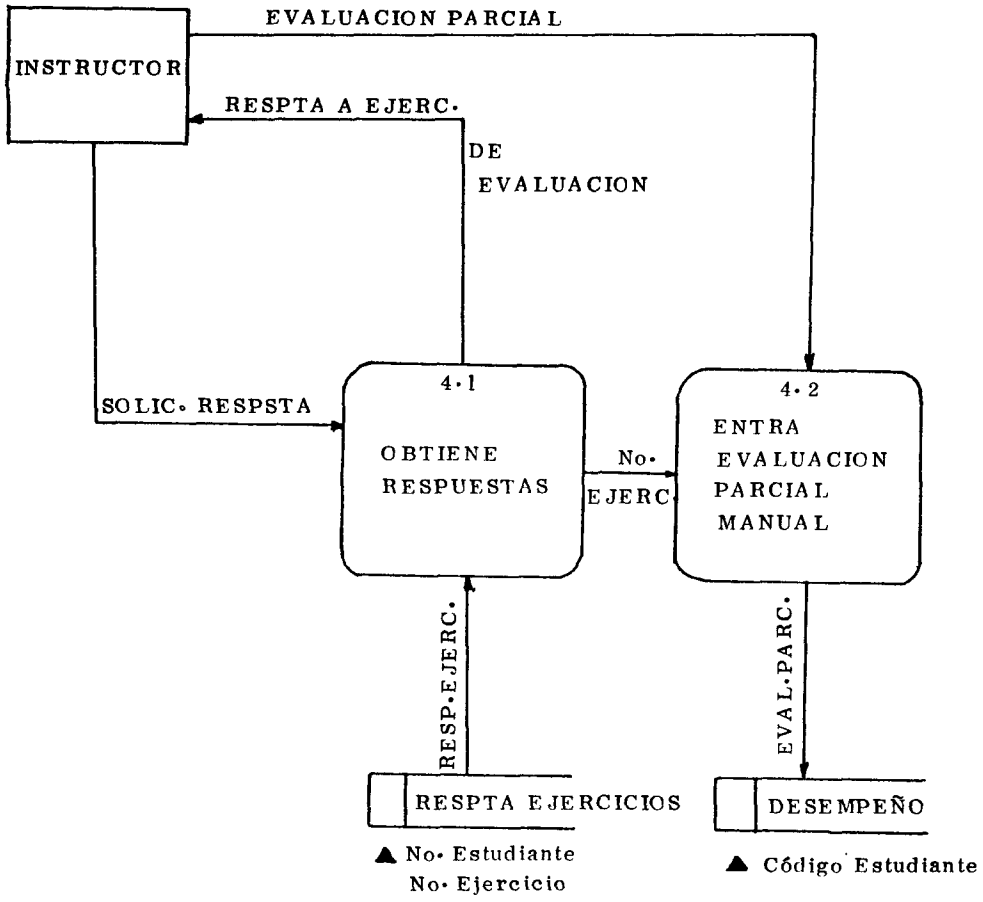


FIGURA 5. Explote 3.0 (Informa Progreso)

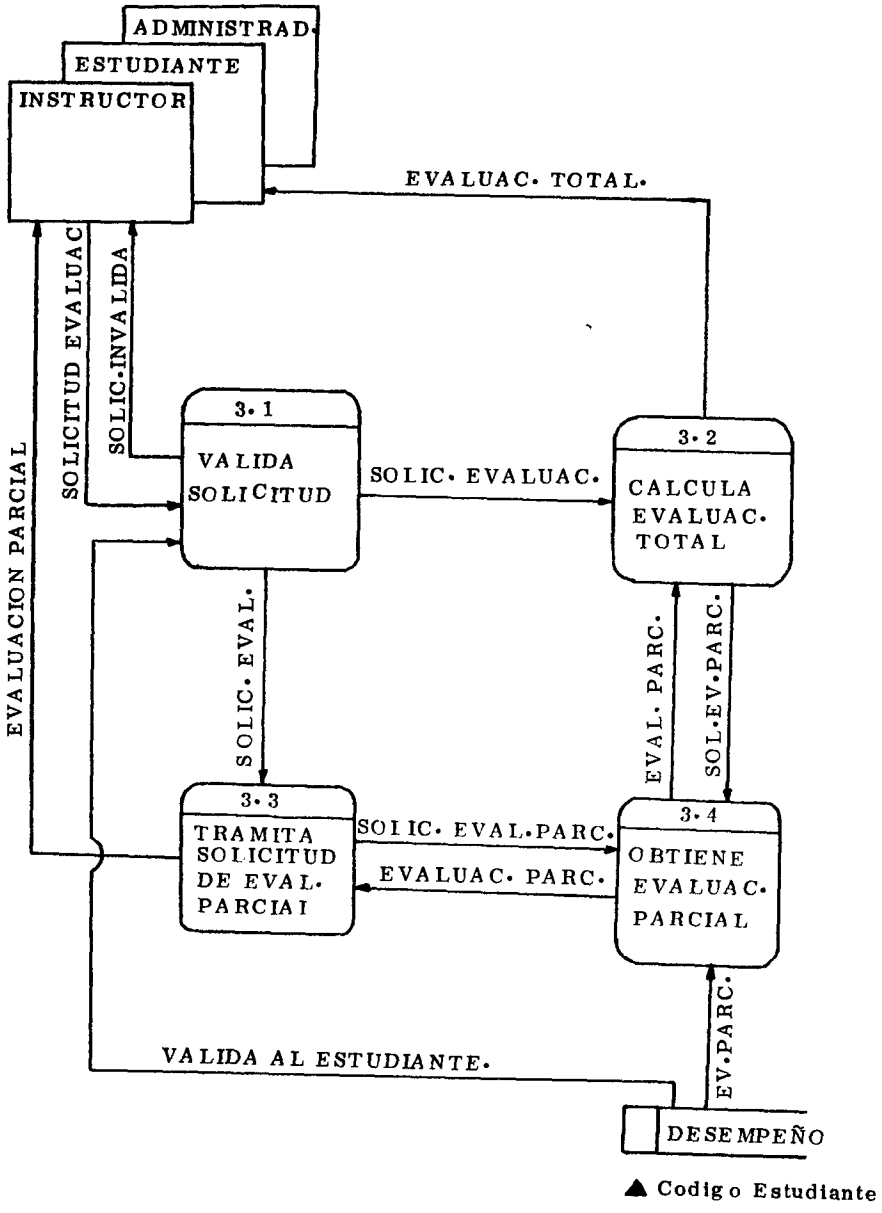


FIGURA 6. Explote 4.0 (Presenta respuestas)

de comandos (Interpretador 5.2).

En la figura 7 podemos ver claramente el funcionamiento del proceso 5.0 y además cómo interactúa con los archivos de desempeño para dar entrada a Notas Parciales que obtenga el alumno en los ejercicios, y, dé respuesta a ejercicios para guardar las respuestas a preguntas abiertas.

Se puede apreciar también que el ingreso del estudiante al sistema por medio del presentador de tramas, se hace únicamente bajo autorización del proceso 2.0 visto en el punto 3.3.2.

3.3.6 Explote 6.0 (Registra Estudiantes)

Este es un proceso de mantenimiento compuesto por dos acciones. Le permite al instructor registrar los alumnos al curso (Da Entrada Datos estudiante 6.2), previa verificación de la no existencia de éste en el archivo de desempeño (Verifica existencia 6.). Ver figura 8.

3.4 DFD'S NIVEL DOS DE PLEAC I

Subprocesos ó subniveles en Pleac I, solo existen para el proceso EXPONE, puesto que en los demás procesos no es necesario mas detalle.

EXPLOTE 5.1 (PRESENTA TRAMAS)

El proceso presenta tramas, se subdivide en dos subprocesos , uno que toma

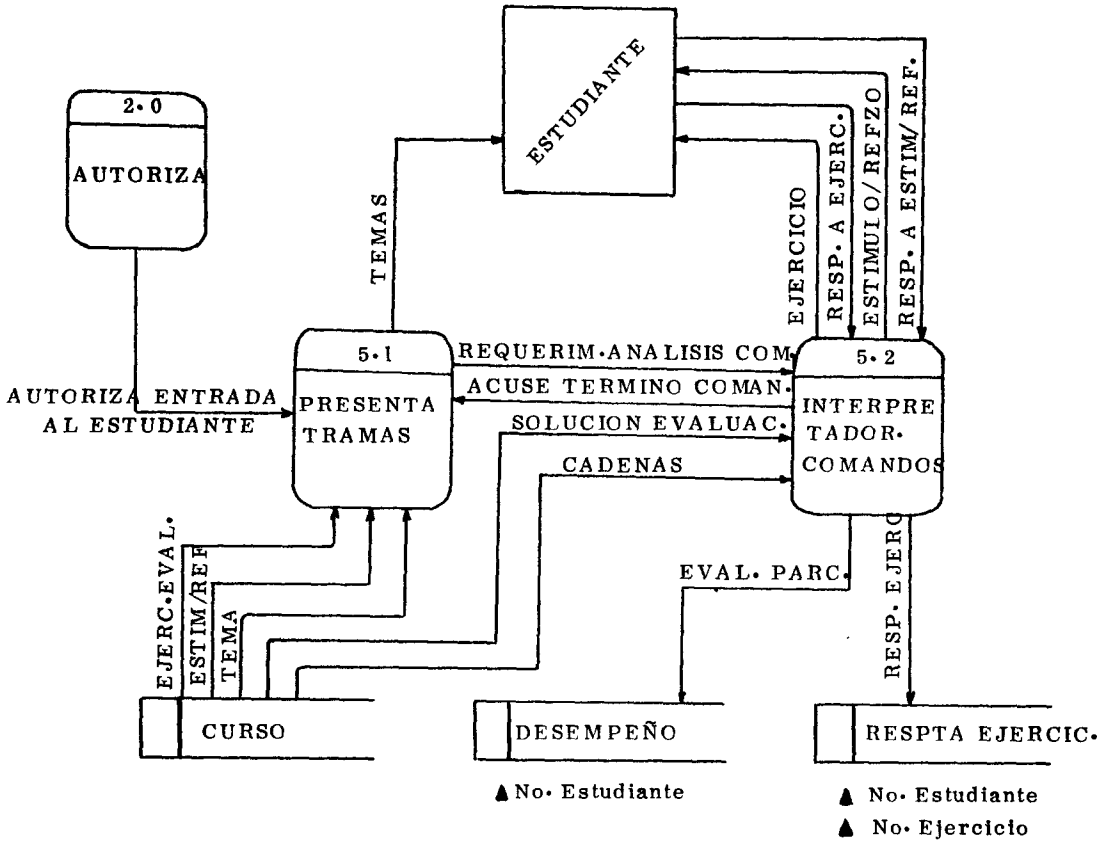


FIGURA 7. Explote 5.0 (Expone)

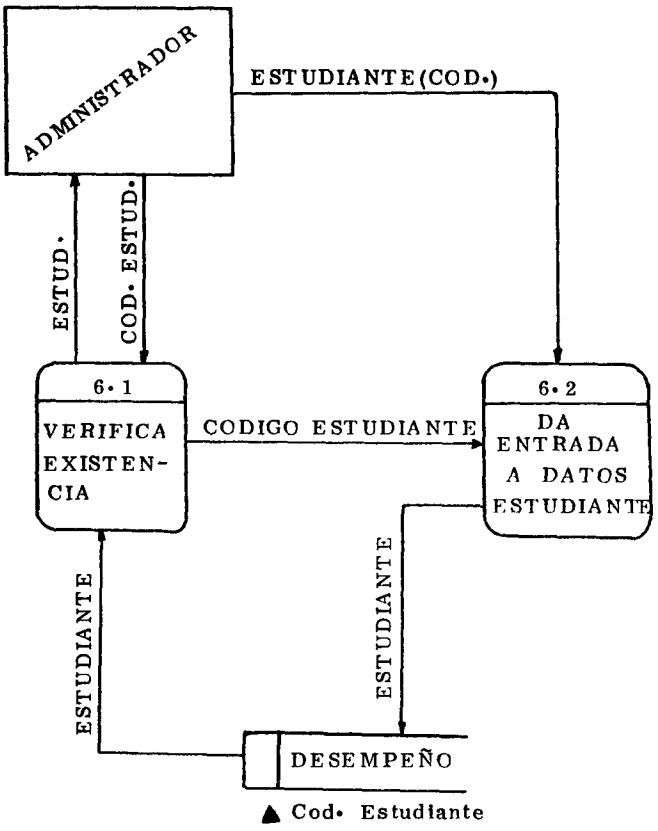


FIGURA 8. Explote 6.0 (Registra estudiante)

los textos del archivo del curso (LEE DEL ARCHIVO DE TEMAS 5.1.1), previa verificación de la autorización que posee el estudiante para ingresar al sistema, y otro que analiza los caractéres de los textos apoyado en el interpretador 5.2 y los presenta al estudiante por pantalla (ANALIZA CARACTERES Y MUESTRA TEMAS 5.1.2), este proceso y la trama de cadenas de caracteres del archivo del curso por parte del interpretador para entregar la interpretación de los comandos al 5.1.2 , la podemos apreciar mejor enla figura 9.

3.5 ESTRUCTURAS DE INFORMACION

Basados en estos diagramas de flujos de Datos, separamos la programación en dos bloques principales: Un programa registro, que contiene los procesos Edita Temas (1.0), Informa Progreso (3.0), Presenta respuestas (4.0) y registra estudiante (6.0) ; y un programa presenta, que contiene los procesos Expone (5.0) y autariza (2.0), que es la llave para que el estudiante ingrese a operar el sistema.

Esta concepción se base en la utilización por parte de los usuarios del sistema. El usuario 1 ó Instructor, necesita utilizar la parte del sistema incluída en el programa Registro, y el estudiante ó instructor 2 necesita la parte incluída en el programa PRESENTA, más aclaramos, no " debe " tener acceso, por razones de seguridad, a la parte que controla el PROGRAMA REGISTRO, a nivel de modificación , aunque si lo tiene a nivel de consulta . Es preciso anotar que ambos programas relacionan constantemente acciones de los 6 procesos, pero básicamente están programados como lo

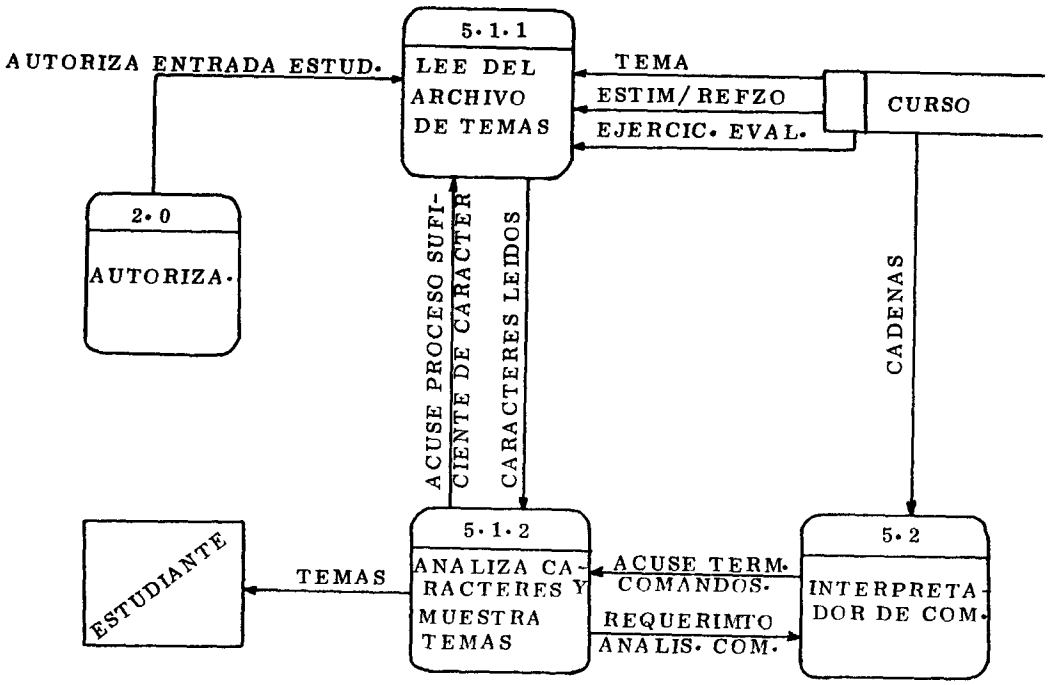


FIGURA 9. Explote 5.1 (Presenta tramas)

anotamos .

Como se puede deducir de lo anterior, el PROGRAMA REGISTRO, tiene opciones de elección para uso del instructor, y su estructura, basada en un menú principal, la podemos observar en la figura 10.

Las figuras siguientes a la 10, muestran el diseño, en forma de árbol, de las rutinas que conforman los programas REGISTRO y PRESENTA.

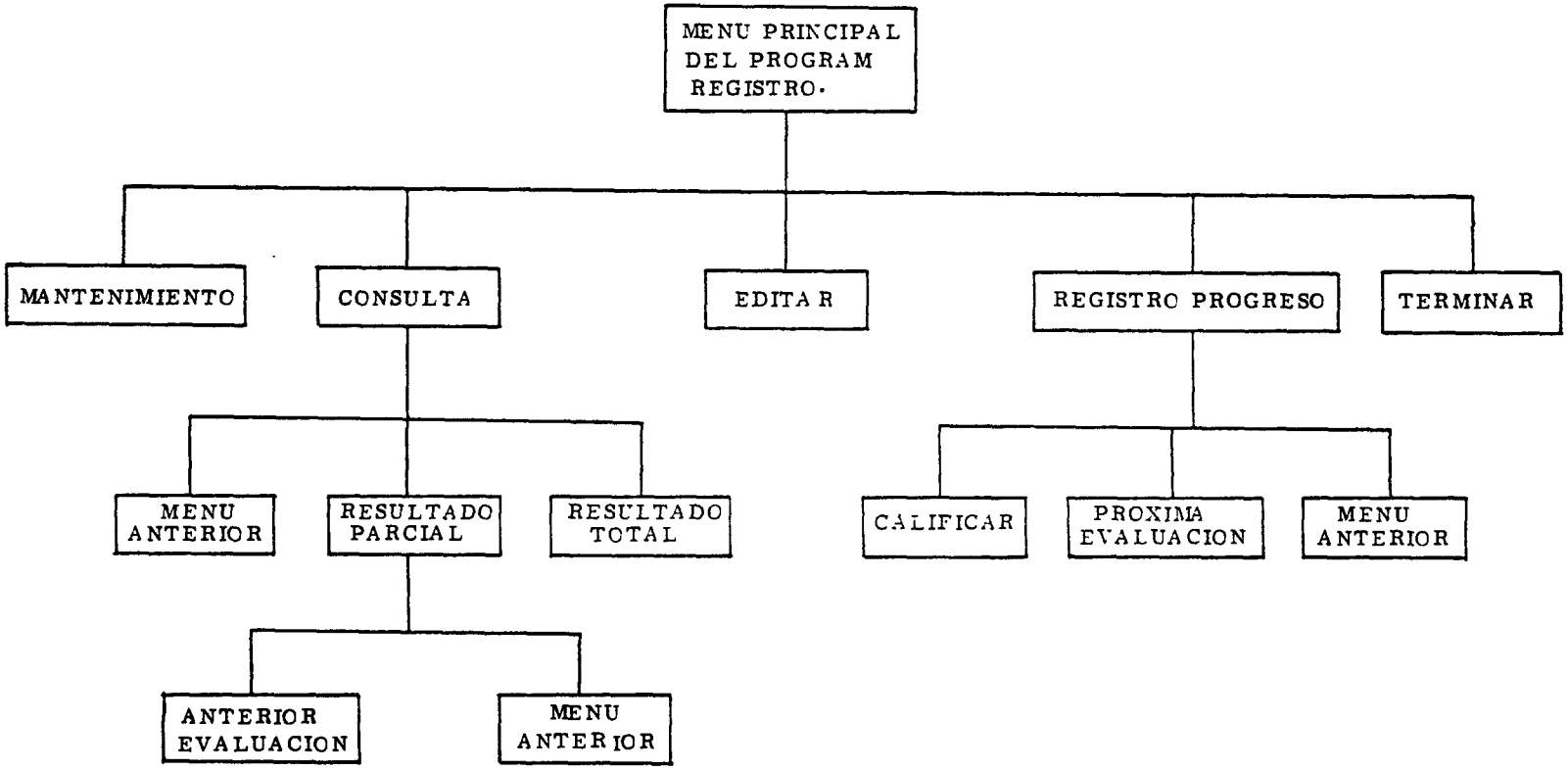


FIGURA 10. Diseño de Menús del Programa Registro

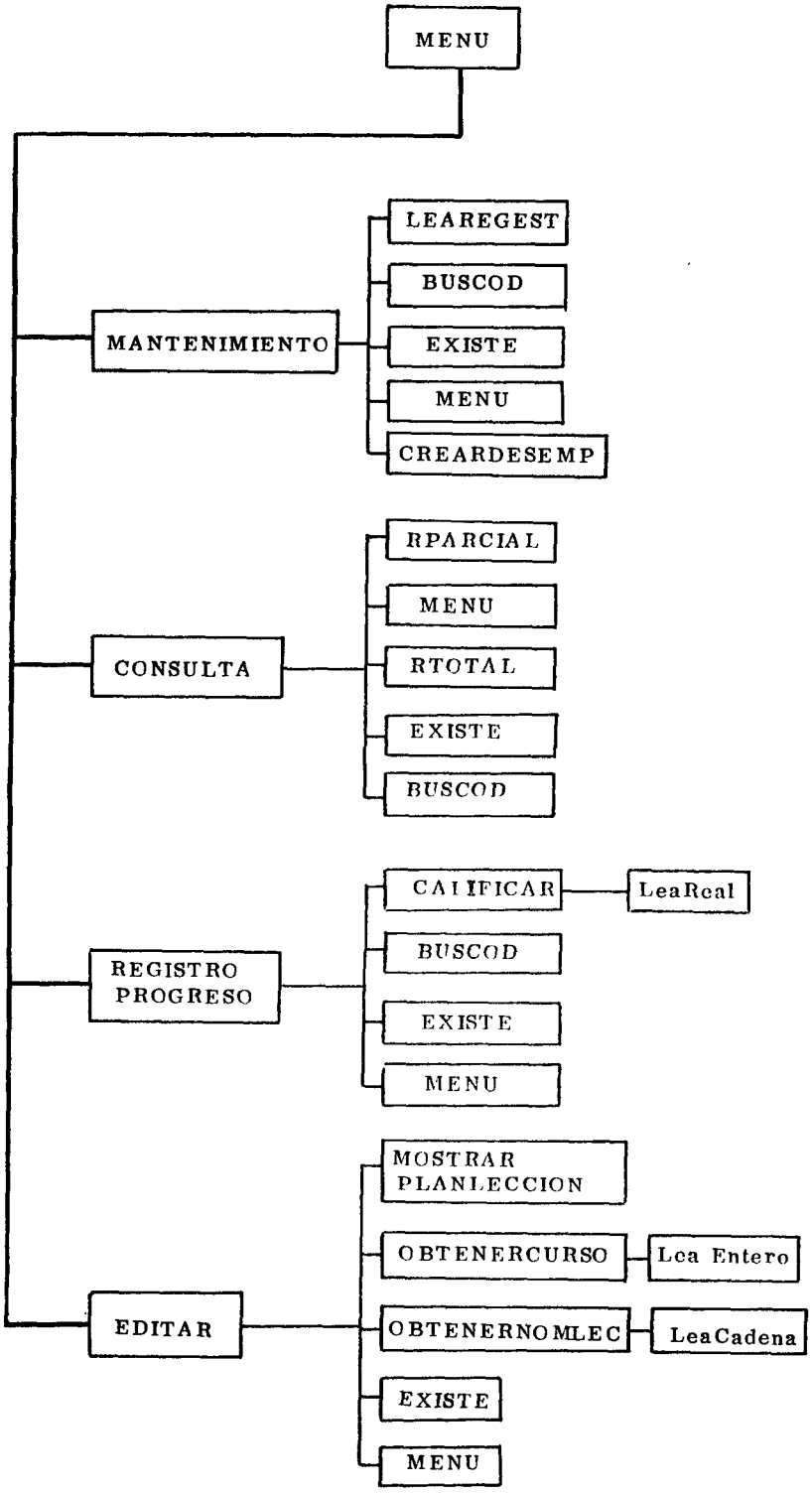


FIGURA 11. Diseño de rutinas del Programa Registro

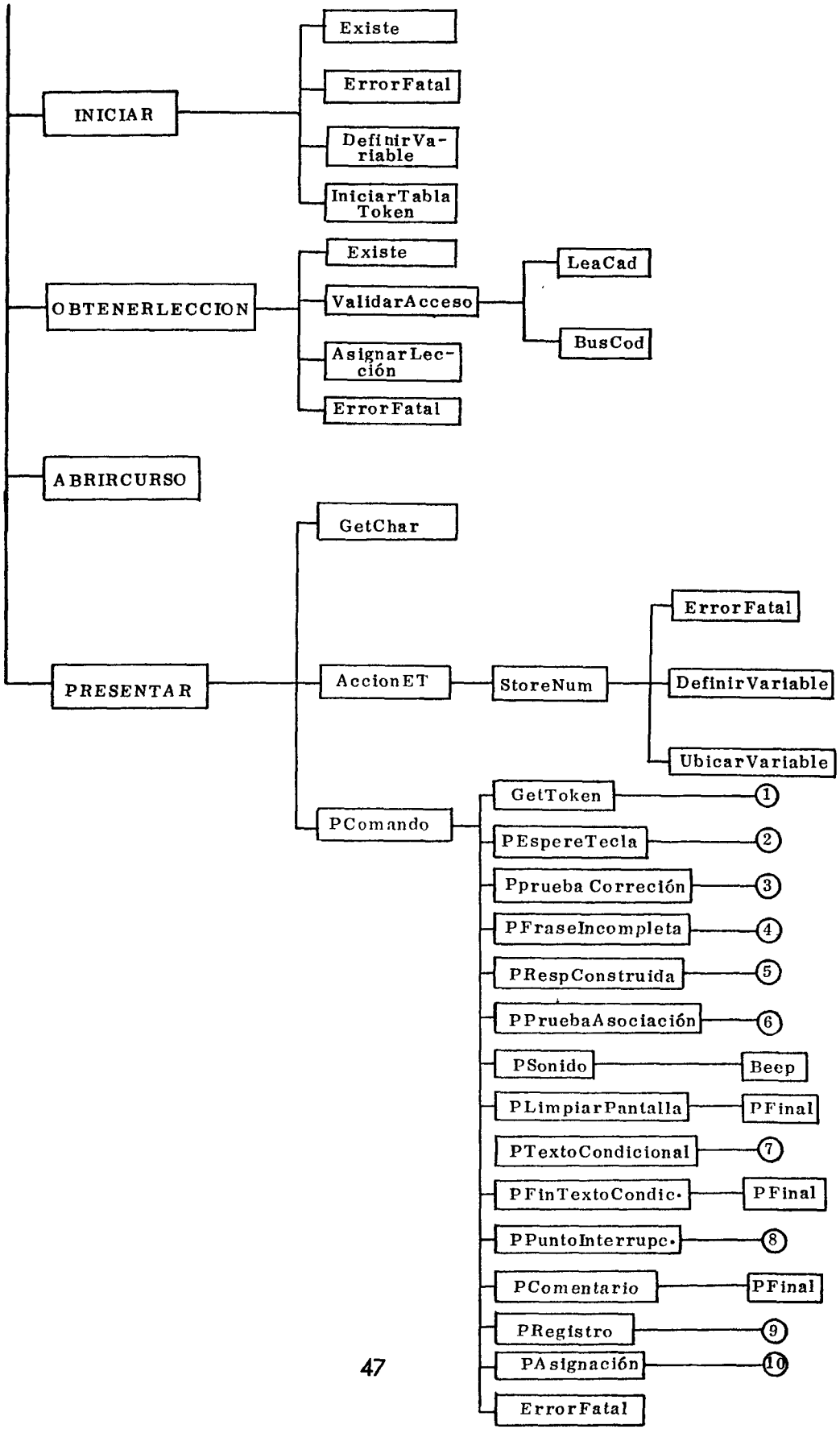


FIGURA 12. Diseño de rutinas del Programa Presenta

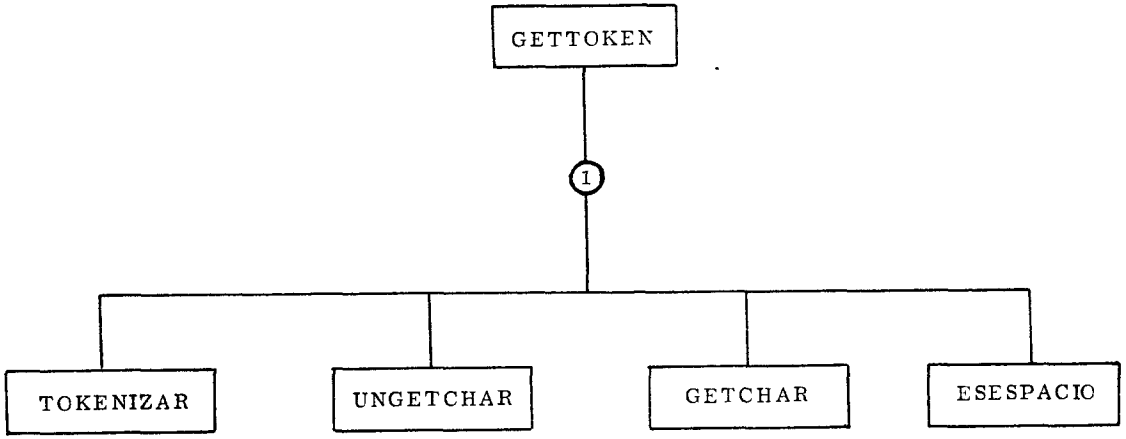
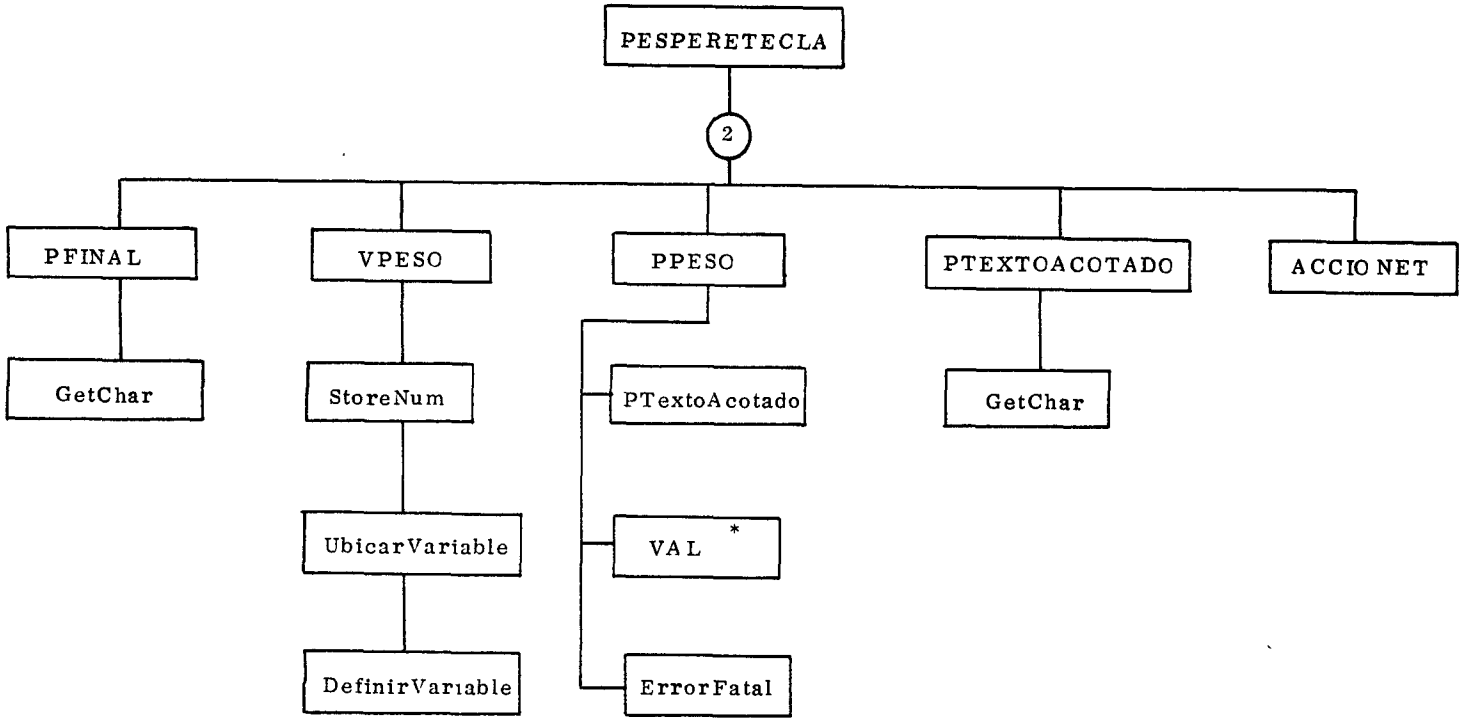


FIGURA 13. Diseño de rutina GetToken



* VAL Es una rutina predefinida en TurboPascal que convierte una cadena de caracteres en un entero.

FIGURA 14. Diseño de la Rutina Pesperetecla

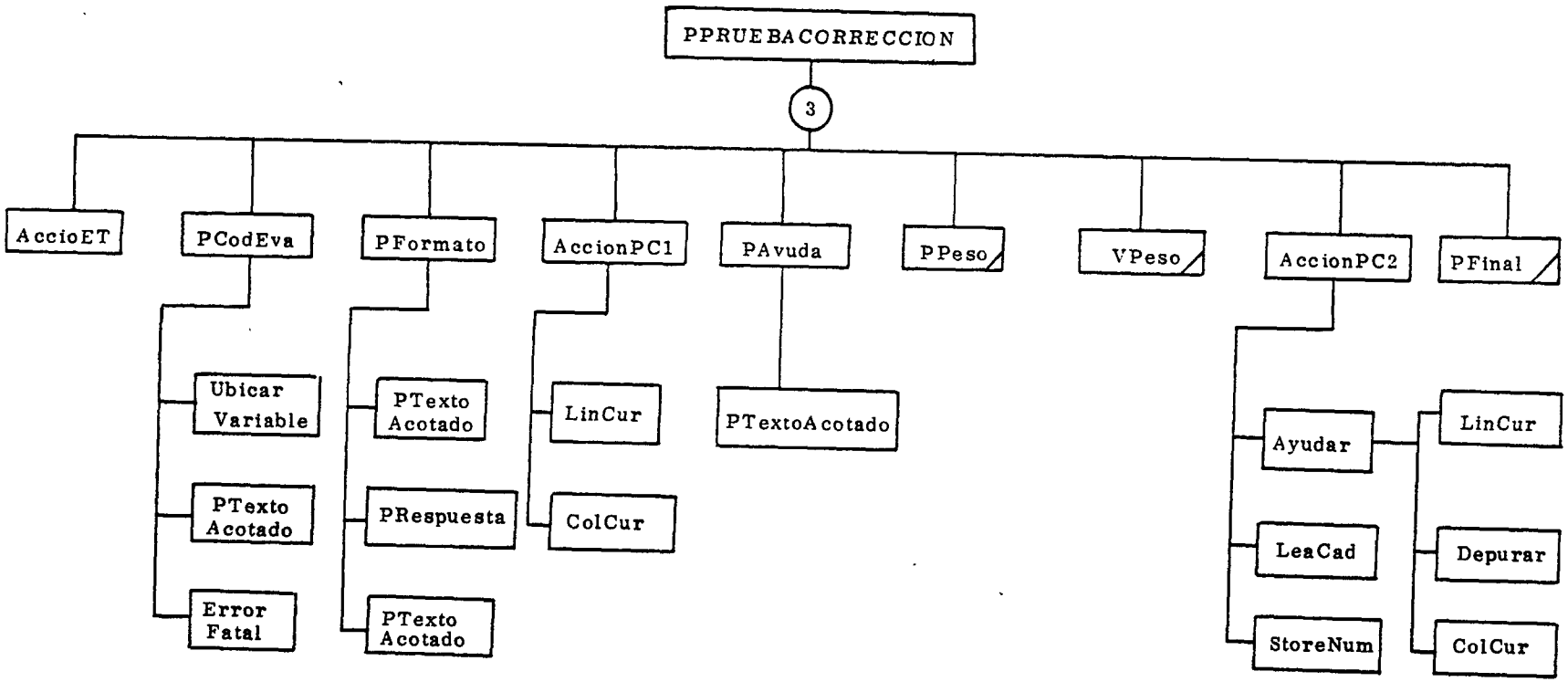


FIGURA 15. Diseño de la rutina PPruebaCorrección

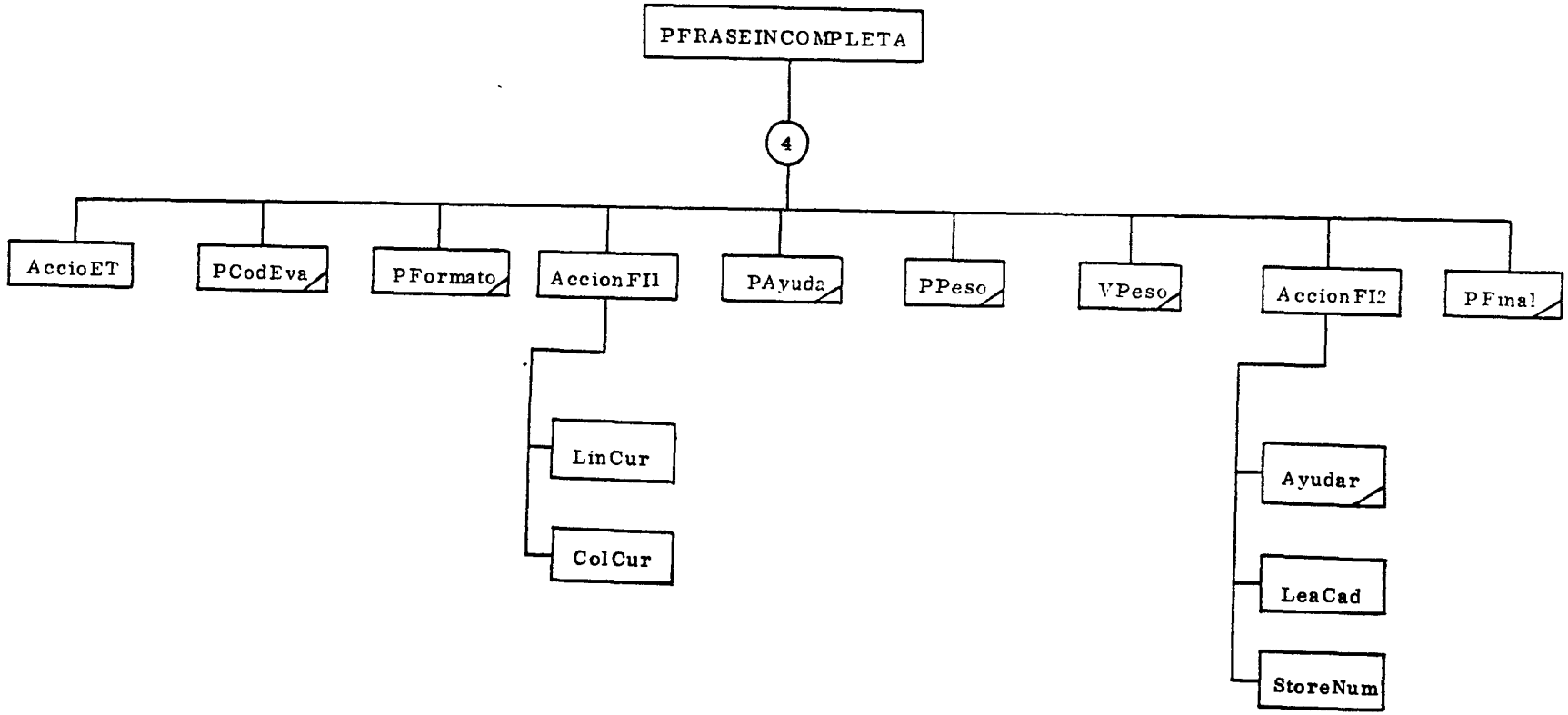


FIGURA 16. Diseño de la rutina PFRASEINCOMPLETA

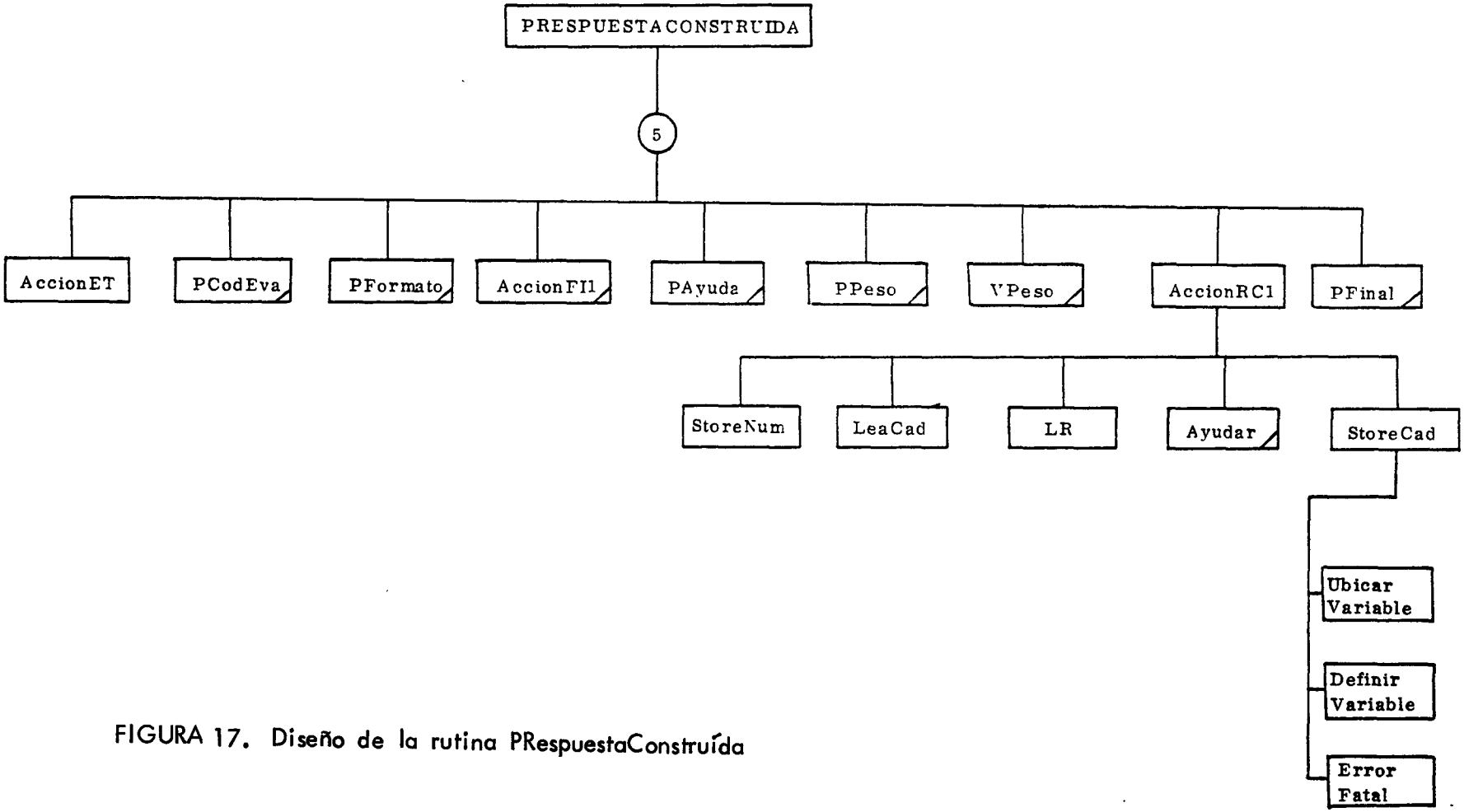


FIGURA 17. Diseño de la rutina PRespuestaConstruída

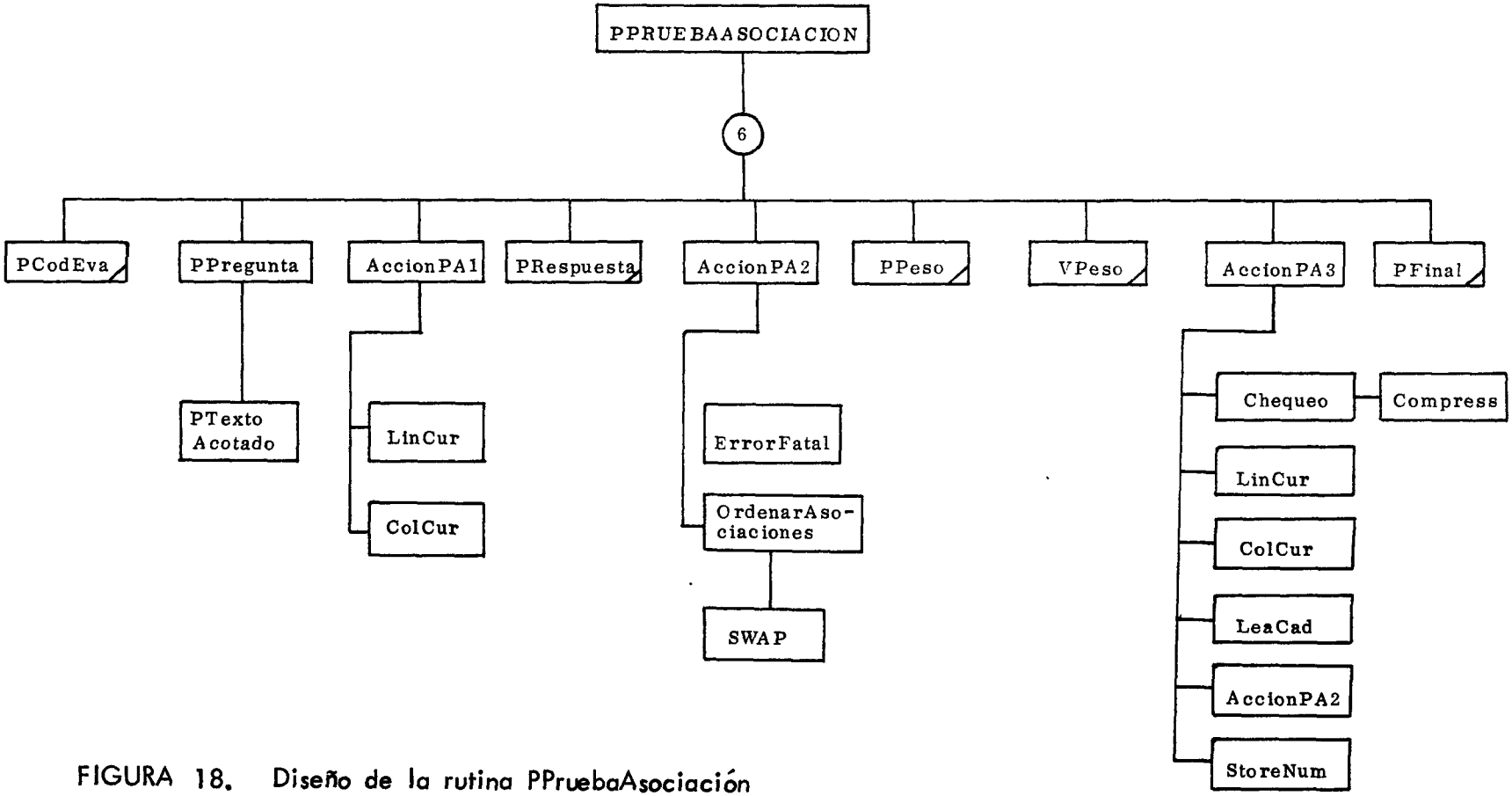


FIGURA 18. Diseño de la rutina PPruebaAsociación

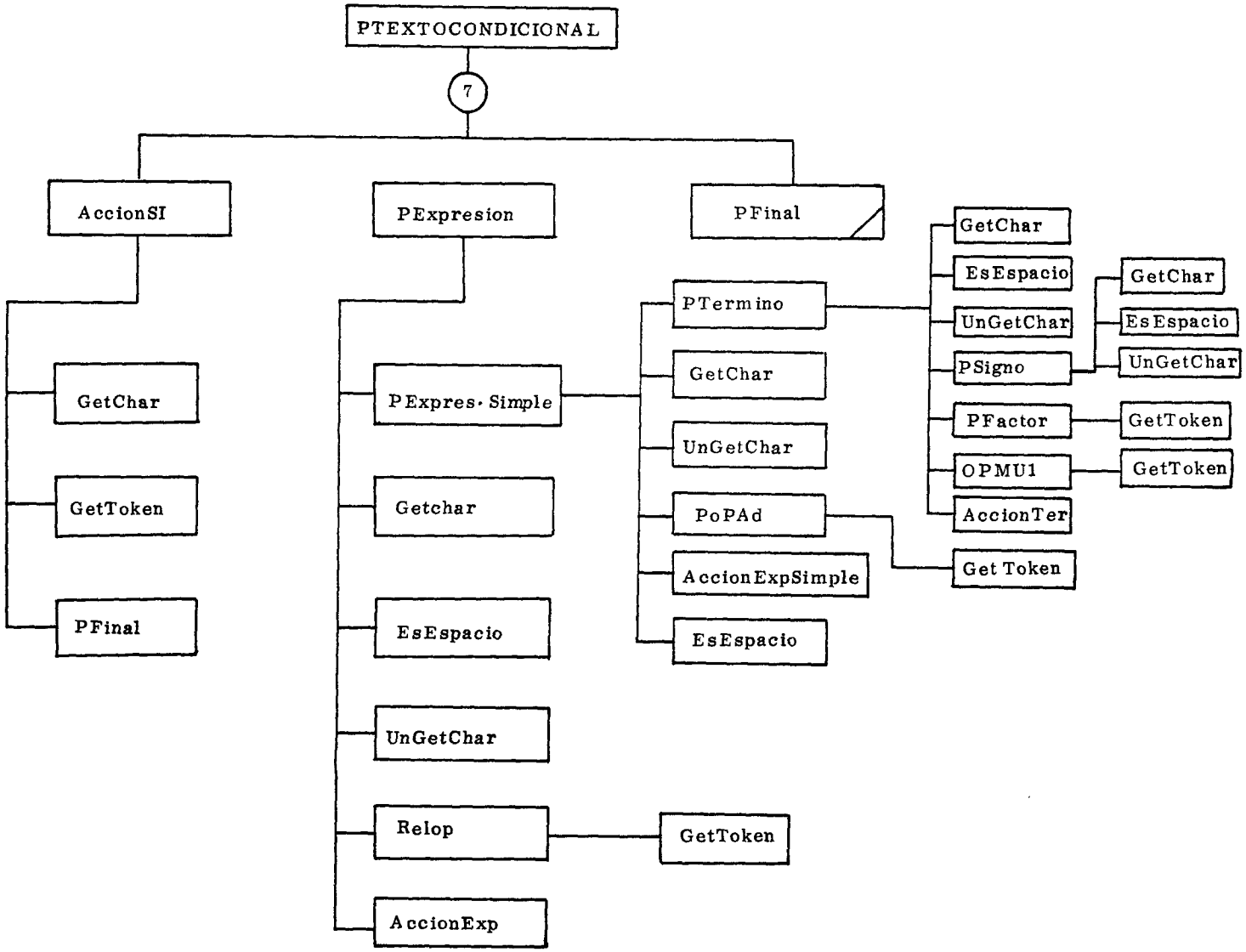


FIGURA 19. Diseño de la rutina PTextoCondicional

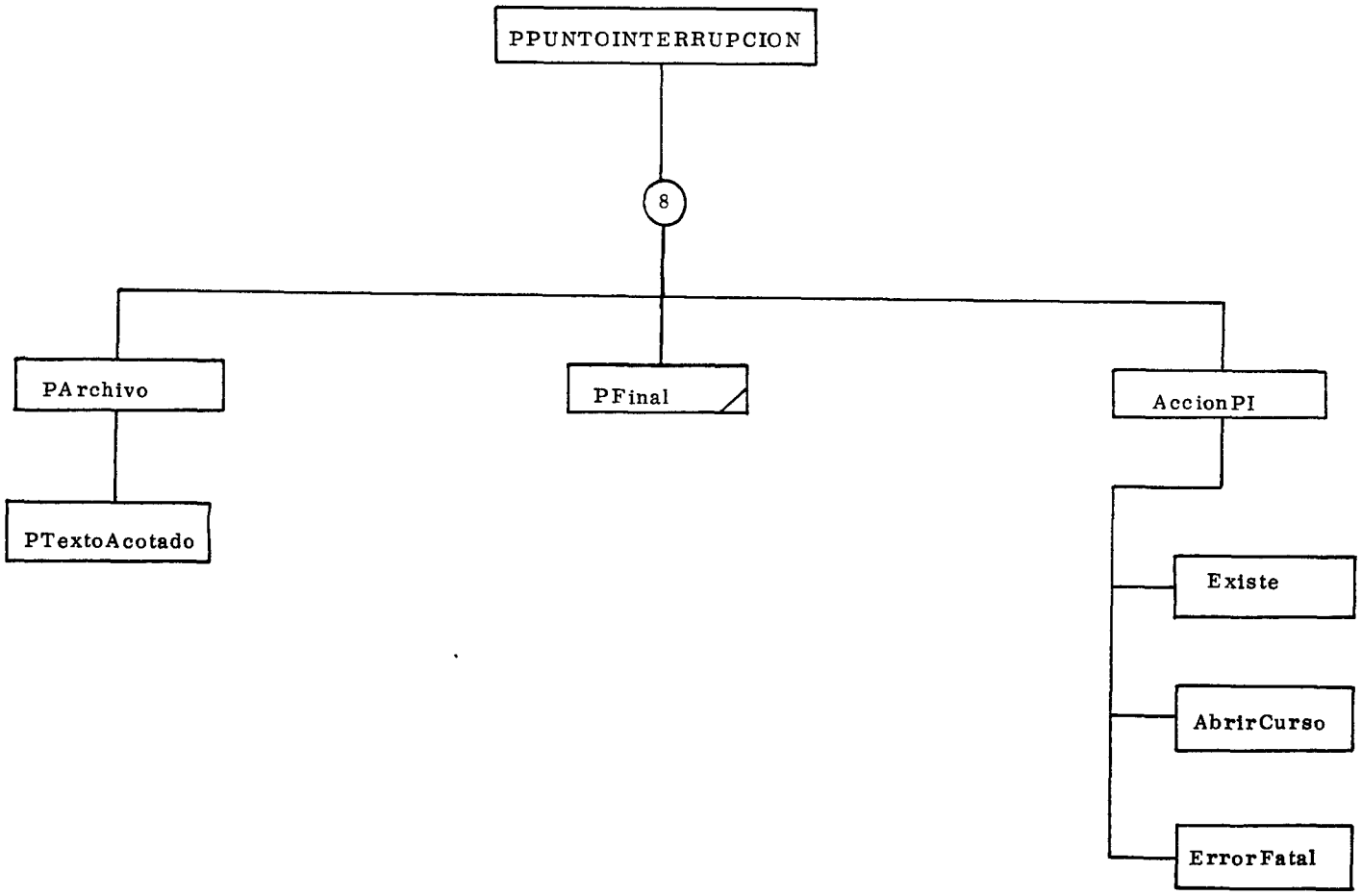


FIGURA 20. Diseño de la rutina PPuntoInterrupción

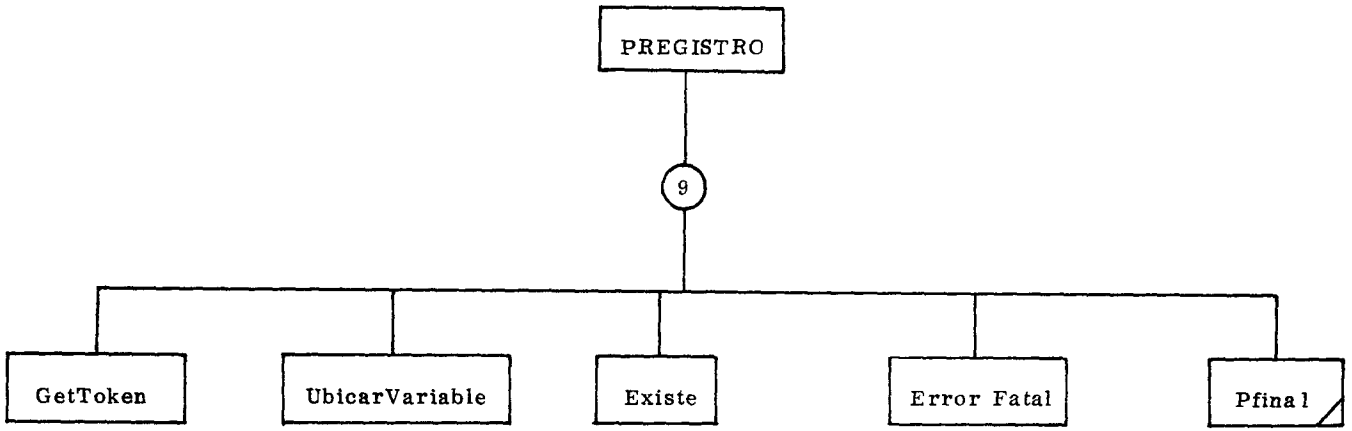


FIGURA 21. Diseño de la rutina Pregistro

4. PROGRAMACION

Este capítulo contiene los programas que se desarrollaron de acuerdo a los diseños que se plantean en el capítulo 3, a la vez que una explicación de las principales rutinas (Funciones y Procedimientos) que conforman los dos programas básicos de PLEAC I: PROGRAM REGISTRO y PROGRAM PRESENTA.

4.1 PROGRAM REGISTRO

Los listados de las rutinas de este programa se muestran en el anexo 1.

4.2 DESCRIPCION DE FUNCIONES Y PROCEDIMIENTOS UTILIZADOS EN PROGRAM REGISTRO

Este programa sirve para que el usuario 1 ó instructor lleve un registro de los alumnos inscritos al curso, de sus evaluaciones parciales y totales. Le sirve además para editar las lecciones que conforman su materia y para consultar y calificar las respuestas dadas por los estudiantes a las preguntas de tipo abierto que haya dejado planteadas.

La información se guarda en tres archivos principales que son :

ARCHIVO DE DESEMPEÑO, que contiene el CODIGO DEL ESTUDIANTE de 6 dígitos , el NOMBRE DEL ESTUDIANTE, con un máximo de 30 caracteres, NUMERO DE LECCION en donde se registra la lección que cursa actualmente , CODIGO DE EVALUACION que es asignado por el Instructor a cada pregunta, PUNTAJE en donde se guarda la nota, PESO ó porcentaje de cada evaluación y otros campos como primer estudiante, próximo estudiante, primera evaluación, próxima evaluación destinados al manejo de colas, ARCHIVO DE RESPUESTAS, en el que se guardan las respuestas de los estudiantes, mostrando el código de evaluación, el peso y el código del estudiante que responde. Se diseñaron otros campos como primera evaluación, próximo y primer libre para control de colas.

ARCHIVO DE LECCIONES, en donde se guardan las lecciones editadas por el Instructor y el nombre de ellas.

El Program Registro está conformado por cuatro rutinas principales que son: MANTENIMIENTO, CONSULTA, REGPROG y EDITAR.

4.2.1 Procedure Mantenimiento

Este procedimiento verifica que exista el archivo de desempeño a través de la función EXISTE. Si no existe propone su creación.

El propósito de este procedimiento es crear un registro de estudiante con dos elementos básicos, el código y el nombre, por lo tanto este procedimiento verifica que no esté ya creado o registrado el estudiante a través

de la función BUSCOD, que valida el código cuando el Instructor lo digita para su creación. Si existe presenta el registro para su actualización, si no existe lo presenta para su adición.

4.2.2 Procedure Consulta

Este procedimiento al igual que el anterior verifica que exista el archivo de desempeño; si no existe envía al Instructor al menú principal, opción mantenimiento, para que lo cree. Si existe, verifica que exista el código del estudiante. Si existe muestra datos básicos del estudiante como el nombre y la lección que cursa. Le dá también la opción al Instructor de consultar resultado parcial o resultado total de notas de los estudiantes.

Si se consulta resultado parcial, se presenta la última evaluación del estudiante así: Número de evaluación, nota y peso y dá la opción para consultar la anterior evaluación o regresar al menú anterior de consulta. Si se consulta resultado total se hacen cálculos de notas totales teniendo en cuenta el porcentaje y las notas parciales y presenta la nota total o el promedio hasta el momento, de acuerdo al porcentaje acumulado y al número de evaluaciones hechas. Si no hay evaluaciones imprime el mensaje correspondiente.

4.2.3 Procedure Editar

Inicialmente verifica que exista el archivo de lecciones, si no existe lo crea dejando un rango de 75 lecciones para ser creadas. Presenta las

opciones de Editar, Suprimir, Insertar, Cambiar, Si se quiere editar una lección, primero se obtiene el nombre de la lección y luego va al editor de texto del DOS para que ahí el Instructor escriba su lección y evaluaciones.

La opción suprimir obtiene el número de la lección a suprimir y luego la suprime corriendo las siguientes un campo hacia arriba. La opción Insertar es con el fín de ubicar una lección entre 2 yá existentes. Primero verifica que exista campo ara insertar , si no existe saca mensaje de error. Si hay campo deja inicialmente un espacio blanco en el número de lección en el cual se quiera insertar el nombre y luego le asigna el nombre dado.

La opción cambiar es con el fin de cambiar el nombre de una lección por otra. Primero obtiene el número de lección (PROCEDURE - Obtener Curso), luego cambia el nombre de la lección por el deseado (Procedure Obtener Nombre Lección).

4.2.4 Procedure Regprog

Este procedimiento verifica que exista el archivo de respuestas, el cual es creado por el estudiante cuando responde a las evaluaciones. Presenta el número de estudiante, el código de evaluación y la respuesta correspondiente a ese código de evaluación. Muestra además la opción de calificar y de ver la próxima evaluación.

Si se quiere calificar, se valida la existencia del archivo de desempeño

y que exista el código del estudiante en éste. Si no existe saca mensaje de error de estudiante número xxx no está inscrito. Luego , si existe el código de estudiante, confronta los códigos de evaluación del archivo de respuestas con el de desempeño. Validada esta información si la respues ta ha sido calificada presenta la neta, si no, dá la opción de calificar en un rango de 0.00 a 5.00 y de dar un peso de 0.00 a 100.00.

4.3 PROGRAM PRESENTA

Los listados de las rutinas que se programaron para implementar este programa se muestran en el anexo 2.

4.4 DESCRIPCION DE FUNCIONES Y PROCEDIMIENTOS UTILIZADOS EN PROGRAM PRESENTA

Este programa sirve para que el usuario 2 o estudiante ingrese al curso para estudiar las lecciones creadas por el instructor y a la vez que responda las evaluaciones dejadas.

Los archivos que se emplean en este programa son los mismos que se describieron en el Program Registro (Desempeño, Respuestas , Lecciones).

El estudiante ve el texto de las lecciones tal como las creó el Instructor. Sin embargo las evaluaciones las ve muy diferentes, en su estructura, como las creo el Instructor, pues éste las elabora de acuerdo a una serie de comandos definidos en un lenguaje que era necesario crear, sin embargo re-

visando los textos bibliográficos encontramos un lenguaje que nos sirvió de referencia , el cual complementamos y adaptamos a nuestro propósito.

Las especificaciones del lenguaje de Comandos de PLEAC, se presentan en el anexo 3.

Para la interpretación de este lenguaje se definió una tabla de comandos llamada T. Token.

El programa presenta , está dividida en 4 bloques principales que son :

INICIAR, OBTENER LECCION, ABRIR CURSO, PRESENTAR.

4.4.1 Función Iniciar

Verifica que exista el archivo de desempeño y el archivo de lecciones para que el estudiante pueda iniciar su curso. Abre estos archivos , luego si no existe archivo de respuestas lo crea e inicializa algunas variables con ceros, también define las variables para nota, respuesta, peso y código de evaluación (CODEVA).

También se inicia la tabla Token, es decir se asignan los significados a cada uno de los Comandos definidos en el lenguaje.

4.4.2 Función obtener Lección

Verifica que esté registrado el estudiante. Si está matriculado asigna el nombre de la lección al campo Archivo del archivo de desempeño de estudiantes, quedando como la lección actual. Si no existe lección actual saca mensaje de error, y escribe fin del curso. Si existe la lección va al paso siguiente, ABRIRCURSO.

4.4.3 Procedure Abrircurso

Este procedimiento abre el archivo del curso que es de tipo text, es decir, un fichero externo de tipo char. Presenta la lección actual al estudiante.

4.4.4 Procedure Presentar

Este procedimiento conforma el cuerpo principal de Program Presenta, pues to que es el encargado de interpretar los comandos y expresiones definidos en cada una de las evaluaciones.

Se van leyendo los caractéres hasta encontrar comandos. Cuando esto ocurre vá al procedimiento PComando para que los ejecute hasta terminar la lección ó hasta que haya punto de interrupción.

El procedimiento Pcomando obtiene las palabras definidas en la tabla T Token para analizar y a cada una le asigna un procedimiento así:

ET : PEspereTecla

PC : PPruebaCorrección

Fi : PFraseIncompleta

Rc: PRespuestaConstruida

Pa: PPrueba Asociación

Pito: PSonido

Lp: PlimpiarPantalla

Si: PTextoCondicional

Finsi: PFintextoCondicional

Pi: PPuntoInterrupción

Co: PComentario

Rg: PRegistro

Variable: PAsignación

La mayoría de estos procedimientos constituyen formas de evaluación y serán explicadas más en detalle en el manual del usuario.

5. MANUAL DEL USUARIO

Este manual está enfocado a dar una orientación sobre el uso de PLEACI, en sus menus diferentes y especialmente a guiar al instructor en la creación de lecciones dándole una serie de recomendaciones y a la vez explicando en forma detallada la construcción de comandos y expresiones para elaboración de evaluaciones. Para este propósito hemos dividido el manual en 4 unidades.

5.1 UNIDAD 1 COMO ENTRAR A PLEACI Y SUS DIFERENTES MENUS

El Instructor debe prender el computador con el diskette del curso. Después de ingresar la fecha y hora aparece un menú de entrada con tres opciones:

La No. 1 para ingresar al Cronograma Registro que es manejado por el Instructor .

La No. 2 para que el estudiante estudie sus lecciones y realice las evaluaciones.

La No. 3 para terminar.

Nos remitiremos especialmente a la opción 1 que al ser seleccionada por el instructor lo lleva a un menú con las siguientes opciones:

Mantenimiento , Consulta, Registro Progreso, Editar, Pw, Terminar.

Para la elección de una opción solo basta digitar la primera letra denotada con mayúscula de la opción elegida y luego dar enter.

5.1.1 Mantenimiento

Esta parte sirve para que el instructor, después de haber ingresado al password (Clave de seguridad), registre los estudiantes que van a tomar el curso, anotando el código (Máximo 6 caracteres) y el Nombre . Si el estudiante no está registrado no puede ingresar al curso.

La clave de seguridad se estableció con el fin de evitar el acceso del estudiante a esta parte.

5.1.2 Consulta

Le permite al instructor revisar si un estudiante está registrado. En caso afirmativo informa de la lección que esta tomando y presenta 2 opciones.

Resultado parcial y resultado total.

Digitando (P) y enter, el programa le mostrará la última evaluación y

le dá la opción de mirar las evaluaciones anteriores digitando (A) y luego enter. De esta manera se pueden revisar todas las evaluaciones del estudiante consultado.

Digitando (T) y enter , PLEACI le mostrará la nota total indicándole el No. de evaluación y peso sobre el cual se ha calculado la nota total.

Del programa registro es ésta a la única parte a la cual tiene acceso el estudiante.

5.1.3 Registro Progreso

Presenta al instructor las respuestas de las evaluaciones de tipo abierto para que las revise y le da la opción de calificarlas. Si ya están calificadas puede modificar la nota si así se requiere. Si desea ver la próxima evaluación puede hacerlo digitando (P) y así revisar el total de evaluaciones pendientes de calificación.

5.1.4 Editar

Lleva esta opción al menú de lecciones donde aparece un formato numerado de 0 a 74 en forma de columna para lecciones y en la parte inferior de la pantalla se dan una serie de opciones , que son :

Edita, Suprimir, Insertar, Cambiar, Menu anterior.

5.1.4.1 Editar (E)

Se selecciona al digitar la letra (E) tal como se especificó antes (Primera letra mayúscula de la opción seleccionada). Se pregunta por el nombre de la lección a editar y el instructor asigna el nombre que desee o que corresponda de acuerdo a su materia . Enseguida aparece el editor de textos para que el instructor programe sus lecciones y evaluaciones . La forma de hacerlo se detallará más adelante.

El editor de textos tiene sus propias instrucciones que el instructor puede revisar y aprender a manejar.

Al terminar de editar, se regresa al menú de lecciones y aparece el nombre de la lección editada.

5.1.4.2 Insertar (I)

No se necesita crear la lección para que su nombre aparezca en el menú de lecciones. El nombre se puede crear con esta opción y luego editarla. Esta opción pregunta por el número de la lección y nombre de la misma y al hacerlo aparece en el sitio indicado del menú de lecciones.

5.1.4.3 Suprimir (S)

Al digitar esta opción se pregunta por el nombre de la lección a suprimir , al hacerlo la lección será borrada.

5.1.4.4 Cambiar (C)

Se hace con el fin de cambiar una lección de ubicación dentro del menú de lecciones.

Se pregunta inicialmente por el nombre de la lección y luego el número o sitio donde se desea ubicar.

5.1.5 Password o Contraseña de Seguridad

Esta contraseña es la llave para entrar a la mayoría de las opciones que maneja el instructor, esta clave es secreta y sólamente debe ser conocida por el instructor debido a que si el alumno la conoce puede entrar a opciones como calificar , editar temas, y a otras que por razones lógicas el estudiante no debe tener acceso.

Este procedimiento específico, permite al instructor cambiar de contraseña cada vez que así lo requiera, para lo cual debe digitar su contraseña antigua o actual y luego la nueva o sea por la que se desea cambiar la actual.

Se sugiere al instructor no digitar su clave delante de otras personas y guardar el máximo de reserva .

5.2 UNIDAD 2. CREACION DE LECCIONES

Toda lección consta de uno o más archivos. Así mismo, toda lección tendrá un archivo principal, por el cual se inicia su presentación.

El archivo principal se crea a través del menú de lecciones con la opción de Editar. Los archivos anexos al principal se crearán por el instructor cuando utilice el comando Punto de Interrupción (PI) el cual será explicado más adelante, en la unidad 4 de este manual.

El instructor podrá controlar el flujo de información a través de la pantalla del computador, interrogar al estudiante , registrar puntajes y respuestas, mediante la inclusión de los comandos de PLEACI.

Todo comando debe empezar después de un punto aparte.

El comando se inicia siempre con un punto, el cual avisa a PLEACI que viene un comando. En la unidad 4 se ilustrará el uso de estos comandos. Sin embargo el siguiente cuadro será útil para la escogencia de un comando según su necesidad.

TABLA 2. Comandos de PLEAC1

Comando	Nombre del Comando	Utilidad
ET	Esperar Tecla	Detener presentación hasta que el estudiante utilice una de un subconjunto de teclas.
PITO	Sonido	Llamar la atención.
PI	Punto de Interrupción	Para permitir al estudiante suspender su lección en un punto determinado.
SI	Texto Condicional	Determinar la presentación u omisión de una sección de texto de acuerdo a una condición definida por el instructor.
FINSI	Fin texto condicional	Para marcar el final de una sección de texto condicional.
LP	Limpiar Pantalla	
RG	Orden de registro	Permite registrar respuestas si despues del comando se digita R o notas si se digita N.
RC	Respuesta construída	Para formulación de preguntas a las que el estudiante debe contestar con una o más palabras o símbolos. Sirve también para preguntas de escogencia múltiple.
PC	Prueba de corrección	Para formulación de preguntas en las que el estudiante debe hacer correcciones.
FI	Frase Incompleta	Para formulación de preguntas de completación.

PA	Prueba de Asociación	Para formular preguntas que requieren más de una respuesta.
CO	Comentario	Para inclusión de comentarios dentro de las lecciones.

Cada vez que termine una lección PLEACI, guarda el nombre de la lección siguiente de tal manera que cuando el estudiante ingrese de nuevo al curso se le presentará la lección posterior a la finalizada en su sesión anterior.

Una lección es básicamente una secuencia de párrafos de dos clases :

1. De exposición

2. De comando

La parte de exposición es la presentación propia del curso y a veces se emplea para el planteamiento de preguntas o ejercicios al estudiante.

Los párrafos de comando pueden ser a su vez de dos clases:

1. Comandos de control de la lección.

2. Comandos para interrogar al estudiante.

Los primeros son los que sirven para avanzar o detenerse en la lección y manejar valores de nota o respuestas del estudiante para su evaluación.

Los comandos para interrogar facilitan la interacción entre la lección y el estudiante.

Como se anotó anteriormente un párrafo de comando se inicia siempre con un punto y sigue una palabra clave que identifica a cada comando.

El siguiente es un ejemplo de párrafo de comando.

PA / PA01 / + La palabra que califica al sustantivo es :

+ ; adjetivo ; & que nos dice una cualidad o característica de él &10.

Estos comendos son una herramienta básica para que en el curso tome participación activa el estudiante y permiten reforzar el aprendizaje y evaluarlo.

En el ejemplo presentado se pueden observar algunos segmentos que necesitan explicación para lo cual se introducen las siguientes definiciones:

Se define con Separador o Delimitador a cualquiera de los siguientes símbolos:

! # $ % & ( ) _ + ( ) : " ? ´ - = / ; . ,

Se define Texto Acotado como una secuencia de cero o más caracteres con tenidos entre dos o más separadores iguales. En el ejmplo presentado se puede observar la palabra adjetivo separada por dos ocurrencias de (;).

Al hablar de cero caracteres hacemos alusión a que pueden programarse

textos acotados vacíos. El siguiente es un texto acotado vacío por dos ocurrencias de separador ($).

$$

5.3 UNIDAD 3. CADENAS, VARIABLES y EXPRESIONES

5.3.1 Cadenas

Una cadena es un caracter o una secuencia de caracteres encerrados en comillas. Por ejemplo: "". Esta es una cadena nula o vacía.

" Z " Cadena de un solo caracter. Longitud 1.

"Las cadenas siempre empiezan y terminan con comillas". Cadena de varios caracteres .

Para PLEAC1 las cadenas pueden tener una longitud entre 0 y 255 caracteres.

5.3.2 Variables

Nos podemos referir a un objeto con un nombre simbólico al que asociamos el valor de ese objeto. Siempre en una variable habrá un valor almacenado , que no se conserva para siempre pues opuede ser cambiado en cualquier momento.

Los nombres de variables deben empezar con una letra y después se pueden combinar letras y números. Ejemplo:

C445d Respuesta

Estos nombres deben ser únicos. No pueden existir dos variables con el mismo nombre.

No se pueden utilizar como variables algunas palabras claves de PLEAC1 como "NO", "Y", "O", como tampoco los nombres de comandos.

Las variables pueden contener : Números , en este caso la variable es numérica , o cadenas , caso en el cual la variable es alfabética.

5.3.3 Expresiones

En una expresión se combinan valores de variables y constantes para obtener resultados numéricos . Este resultado se puede almacenar en una variable o se puede utilizar para reconocer alguna situación en la lección.

PLEAC1 reconoce 3 clases de expresiones:

1. Expresiones aritméticas

2. Expresiones relacionales

3. Expresiones lógicas

En una expresión aritmética se combinan constantes numéricas y variables numéricas para obtener valores numéricos. Ejemplo:

2 * 2

C +F-8 (si C y F son variables numéricas)

8+C /10

El signo "+" especifica la suma, el " - " la resta, el "*" la multiplicación y el "/" la división. Cada uno de estos signos es un "OPERADOR" Debido a que especifican operaciones aritméticas decimos que son "Operadores Aritméticos".

Exite un esquema standar llamado "Reglas de Precedencia" para determinar el orden en que se realizan los cálculos para evaluar la expresión. Las operaciones se efectúan de izquierda a derecha y los operadores "*" y "/" tienen más alta precedencia que los signos "+" y "-", es decir, primero se efectúan las operaciones de multiplicación y división. El orden se puede cambiar utilizando paréntesis que tienen la mayor precedencia. Ejemplos: a+c/5.

(a + c) / 5

Si a = 5 y c = 10 , la primera expresión tiene un valor de 7 y la segunda un valor de 3.

Los operadores relacionales ayudan a comparar dos valores para ver que relación existe entre ellos, para lo cual nos valemos de los siguientes signos:

" " Mayor que

" " Menor que

" = " Mayor o igual que

" = " Menor o igual que

" " Diferente de .

Si la relación establecida es cierta, la expresión relacional tiene un valor diferente de cero. Podemos asociar el valor de cero con falso y cualquier otro.

Si la relación establecida es cierta , la expresión relacional tiene un valor diferente de cero. Podemos asociar el valor de cero con falso y cualquier otro valor con cierto.

Estos operadores son evaluados de izquierda a derecha y tienen igual precedencia. Ejemplos:

6 = 12 - 6 Resultado cierto

6 15 Resultado falso (0)

20 = 20 Resultado cierto

"cc" "CC" Resultado falso (0)

"Batidor" "Burro" Resultado cierto

Se puede observar que podemos comparar números o cadenas, pero siempre objetos del mismo tipo.

El resultado de una expresión relacional es numérico.

Los operadores Lógicos, permiten tomar decisiones al igual que los relacionales, pero con condiciones más complejas.

PLEAC1 reconoce tres operadores lógicos:

"y", "0", "NO".

En la relación "0", con que se cumpla una de las condiciones la relación es verdadera.

Si c = 5 y d = 8, entonces:

(c 8) o (d 10) es cierto

(c 3) o (d 9) es falso.

En la relación "Y" se deben cumplir las dos condiciones para que sea cierta.

Tomando los mismos valores de "c" y "d" del ejemplo anterior, tenemos:

(5 c) y (8 = d) es falso

(c 3) y (d = 8) es cierto.

El operador "NO" indica una operación contraria.

Siguiendo con los mismos valores de "c" y "d", tenemos:

No. (5 8) es cierto

No (c = 5) o (d 6) es cierto

No (d 15) es falso

Los operadores lógicos tienen también su precedencia. Primero se resuelven los "No", luego los "Y" y finalmente los "O".

Cuando una o más expresiones relacionales se conectan mediante el empleo de operadores lógicos, las expresiones relacionales se deben encerrar en pa réntesis.

Existen dos variables predefinidas por PLEAC1 ellas son :

NOTA: Variable numérica en que se guarda el número de intentos empleà do por un estudiante para alcanzar la respuesta en una prueba determinada.

Respuesta: Variable alfabética en que se registra la última respuesta dada por el estudiante.

5.4 UNIDAD 4. COMANDOS

En la Unidad 2 se especificó que para que el instructor pudiera programar sus evaluaciones requería del uso de comandos. En esta unidad explicaremos como crearlos .

Los comandos reconocidos por PLEAC1 son:

Comando	Significado
.et	Esperar tecla
.pito	Pito
.lp	Limpiar pantalla

.si	Texto condicional
.Finsi	Final texto condicional
.pi	Punto de interrupció
.pc	Prueba de corrección
.fi	Frase incompleta
.pa	Prueba de Asociación
.co	Comentario
.rg	Registrar progreso
.rc	Respuesta construída

5.4.1 Explicación / Recomendaciones sobre el uso de PLEAC1

El contenido del curso se presenta en texto común y corriente, como ocurre en este párrafo.

Usted debe tener en cuenta algunas recomendaciones:

Nunca escriba explicaciones de más de una página.

Si lo hace, estas serán muy largas para el estudiante.

Al terminar cada pantalla, para detener la presentación mientras el estu-

diante la lee, utilice el comando ESPERAR TECLA, como se hace a continuación.

Para continuar presione la tecla ENTER.

.et / //0.01/ ═══ El ENTER puede formar parte de un texto acotado ═══.

Al continuar después de que el estudiante avisa que terminó de leer la pantalla, limpie la pantalla con el comando LIMPIAR PANTALLA.

Usted puede interrogar al estudiante en una de las siguientes modalidades:

1. Prueba de corrección (comando. PC)

2. Frase incompleta (comando .FI)

3. Prueba de asociación (Comando .PA)

4. Texto condicional (Comando .SI)

5. Final texto condicional (comando .FINSI)

6. Respuesta construída (Comando .RC)

La prueba de corrección (.PC), consiste en la presentación de una

afirmación (o frase) al estudiante, dentro de la cual hay una porción incorrecta. El estudiante deberá corregirla.

Por ejemplo, en un curso de Ortografía , se podrá plantear una prueba como la que se muestra a continuación:

.pc /PC01 / /Pedro y María se // cazan/ /casan/ /CAZAN, es un término que hace referencia al ejercicio de perseguir y atrapar una presa . Pedro y María contraerán matrimonio.

Usted debe utilizar el fermino correcto. / 10/ ═══ Uso del formato : frase incorrecta correcta

La frase incompleta (.FI), consiste en la presentación de una afirmación (o frase) de la cual falta una porción, preferiblemente , clave para la expresión de concepto . El estudiante deberá completar la frase rellenando un espacio destinado para tal fin.

Observe el ejemplo a continuación:

.fi

/FI01 /

/ Una frase incompleta consiste en mostrar una frase en la que falta una porción , que el estudiante deberá suministrar mediante la asociación de

la misma a un / / concepto / /.

(Escriba su respuesta en minúsculas) / / Recuerde : Lo importante es que el estudiante capte el CONCEPTO, no que memorice una siere de PARRAFOS.

/ 10 / ======= OJO . . . . Si el estudiante contesta en mayúscula y correcto, no pasa.

La prueba de asociación (.PA), consiste en la presentación al estudiante de preguntas que requieren mas de una respuesta, sujeta a la condición de que todas las respuestas sean de 4 o menos caracteres.

La pregunta se plantea como parte del texto normal del curso (no dentro del comando . PA).

Un ejemplo típico, es en el que el estudiante , debe formar parejas con los índices de los listas que contienen entradas relacionadas.

El estudiante deberá escribir sus respuestas, separadas únicamente por comas: Ejemplo:

Cuáles son las notas musicales?

El estudiante deberá contestar :

DO, RE, MI, FA, SOL, LA,SI

No importa si sus respuestas estan en mayúscula o minúsculas o si están en otro orden. Podría haber contestado: MI, la, DO, FA,RE,SOL,SI y sería correcto.

La prueba de asociación admite una frase previa a la lista de respuestas. Esta será presentada al estudiante como solicitud de respuestas:

Ejemplos:

Sin la frase:

Cuáles son las notas musicales ?

_____ .pa

/PA01 //// Do,Re,Mi,Fa,Sol,La,Si // 10 / jbhrtguk

Con la grase :

.pa / PA02 // Las notas musicales son / / Do,Re,Mi ,Fa,Sol,La,Si //10 /jbhrtguk .1p

Un ejemplo final:

Establezca relaciones entre las dos listas a continuación:

1. Mecánica	a. Números
2. Músico	b. Trucos
3. Matemático	c. Automóviles
4. Magos	d. Melodías

.pa

/PA03/

/Cada relación debe separarse de la siguiente con una coma, cada relación consiste de un número (lista izquierda) seguido de una letra (lista derecha), / $1c,2d,3a,4b$ 10 ========= jhfghrewjhf oirtugoirtuhoi63ru9o ===

La respuesta construída (.RC) se utiliza para que el estudiante construya una respuesta, como su nombre lo indica.

El formato esta compuesto del comando seguido de 6 textos acotados obligatorios que son en su orden:

Código de evaluación + texto 1 + respuesta + texto 2 + ayudas + peso. Texto1, respuesta y texto 2 son tomados por el. programa como una frase, de la cual sólo se presenta al estudiante texto1 y texto2 separados por un espacio suficiente para completar la frase. El estudiante puede avanzar al segmento siguiente de la lección si contesta la respuesta correcta, que será la registrada por el instructor. Se recomienda que la preguntas produzcan respuestas precisas no sometidas a dualidades.

Si el texto correspondiente a la respuesta es vacío no se efectúa chequeo sobre la respuesta del estudiante. Esta se almacenará en el archivo de respuestas y posteriormente el instructor las podrá revisar y calificar.

.rc /RC01/ /Qué opina de las bebidas dietéticas?/

///////10/ == Ejemplo donde el texto correspondiente a la respuesta es vacío =

.RG R

Con el anterior comando se registra la respuesta para la revisión posterior del instructor.

También se puede emplear para preguntas de escogencia múltiple, pero planteando la pregunta como parte del texto.

La Novela "Cien años de Soledad" fue escrita por :

a. Garcilazo de la Vega

b. Dante Alighieri

c. Gabriel García Márquez

d. Homero

e. Ninguno de los anteriores

.RC /RC02 / / Su respuesta ? // c/////10/ = El estudiante seleccionará la respuesta ====

Es importante anotar que existe una variable NOTA en donde se guarda el número de intentos generados por el estudiante para alcanzar la respuesta correcta. El instructor puede utilizar esta herramienta para definir la calificación a cada una de las preguntas de evaluación que presente en el curso.

.RG R

.RG N

El anterior comando (.RG), se utiliza para registrar el progreso del estudiante , en un archivo creado al iniciar la lección, en donde se guarda las respuestas del estudiante si se asigna una (R) después del comando RG

o se guarda la nota si en lugar de la (R) se escribe (N).

Siempre que se quiera guardar la respuesta o la nota se debe usar este comando de lo contrario el instructor no podrá visualizar en el archivo de progreso las respuestas que los estudiantes han dado a cada una de las evaluaciones .

Si el instructor desea registrar la nota del estudiante para o preguntas que no son abiertas, se puede valer de otro de los comandos de PLEAC1 que es el de Texto Condicional (.SI) que va seguido de una expresión , que es otro de los elementos importantes del lenguaje, y termina con un final de Texto condicional (FINSI).

Para lograr la calificación el instructor se puede valer de una de las variables definidas por el usuario como es NOTA en la cual se guarda el número de intentos generados por el estudiante para lograr la respuesta correcta y buscar fórmulas sencillas para que el computador le asigne la nota correspondiente . Veamos un ejemplo:

.FI /F102 /

/La expresión (4*5) + 3 *8) da : / /44/ / ./

/Recuerde que lo primero que se resuelve en las expresiones matemáticas son los paréntesis / / 10/.

Se hace uso aquí de expresiones dentro de una frase incompleta.

.RG R = Se registra la respuesta =

.Si NOTA 4;

.NOTA: = 4;

.FINSI

.NOTA: = 4.5 - (NOTA - 1) (4.5 - 1.5) / (4-1);

. RG N

Se ha tomado como referencia que el máximo número de intentos que podrá utilizar el estudiante para hallar la respuesta es de 4 y que si genera mas de 4 el programa siempre tomará como referencia 4. Los números 4.5 y 1.5 son los límites superior e inferior de la nota. Con la última operación lo que se hace es convertir NOTA en una calificación que se guardará en el archivo de desempeño del estudiante.

PLEACI tambien permite que el profesor pueda utilizar variables y asignar le valores mediante el comando Asignación que tiene la forma general:

.Variable : = Expresión ; El comando debe terminar con (;)

.a = (2*5) + 16/ (2*4); Observe que va (;) al final del comando.

En este caso a la variable (a) se le asigna el valor de la expresión de la derecha, es decir, 12. También se puede hacer con cadenas.

.a:="CARBONO"; La cadena asignada debe ir entre comillas.

Usando el anterior comando, sugerimos como fórmula para que el computador evalúe las preguntas que exijan respuestas del estudiante, la que plantearemos a continuación con un ejemplo:

.FI#FI03#

(4*8)-3+ (6/2) da :

/32//./

Empiece resolviendo los paréntesis y luego de izquierda a derecha los signos * y

/ 10/

.MAXI :=3; Máximos Intentos

.MINI:=1; Intentos Mínimos

.MAXN:=5; Máxima Nota

.MINN: =1; Mínima Nota

.SI Nota MAXN; .NOTA:=MAXN; .FINSI

.Nota := MAXN - (Nota-1)* (MAXN-MINN) / (MAXI -MINI); Fórmula

.RG N Finalmente se registra la nota.

Cuando el instructor quiera llamar la atención sobre alguna parte importante del curso puede hacer uso del comando SONIDO (.PITO) que emite un pito, y se puede agregar un comentario opcional después del comando . Veamos:

.PITO Ponga especial atención a esta parte de la lección.

Si el instructor quiere incluir algún comentario dentro de la lección y desea que sea ignorada por el estudiante puede hacer uso del comando COMENTARIO (.CO) Ejemplo:

.CO si el estudiante no contesta lo correcto no avanzara en el curso.

Finalmente cuando el instructor finalice una lección o cuando considere que es tiempo suficiente de estudio de una unidad puede sugerir al estudiante que interrumpa valiéndose del comando PUNTO DE INTERRUPCION (.PI)

adicionando al final opcionalmente un comentario, que sera ignorado por PLEAC1, será solo una anotación para el instructor más no para el estudiante, pues después del punto de interrupción siempre se termina el archivo de la lección actual. Enseguida Pleac1 pregunta al estudiante si desea continuar o no. Si la respuesta es afirmativa PLEAC1 utiliza la primera lección registrada en lecciones para continuar. En caso contrario se suspende la lección y cuando el estudiante reinicie su estudio el programa lo llevará automáticamente al lugar donde interrumpió su lección. Veamos un ejemplo:

.PI /PLEAC1. L02 / Observe que después del comando va un texto acotado con el nombre del archivo donde se reanudará la lección y éste deberá ser el utilizado por el instructor para continuar con el curso.

6. CONCLUSIONES

Después de haber desarrollado y probado el programa PLEAC1 de acuerdo a nuestros objetivos planteados inicialmente, podemos llegar a la siguiente conclusiones:

El lenguaje Turbo-Pascal es un lenguaje de alto nivel que permite elaborar programas autores y por lo tanto nos sirvió para crear el nuestro en el área educativa.

Como era nuestro propósito inicial se pudo crear una nueva herramienta de ayuda a través del computador, en el área de enseñanza - aprendizaje, permitiendo que el profesor pueda generar sus contenidos programáticos y sus evaluaciones en el microcomputador para que el estudiante los consulte y estudie.

El programa PLIAC1 es una gran ayuda en las labores administrativas y académicas del profesor, lo cual le da tiempo para que haga un seguimien to más personal de los estudiantes, para que haga más investigación y así vaya mejorando los programas de las materias que enseña.

A través de la elaboración del programa PLEAC1 se pudo ver como éste

abre una puerta al estudiante, pues no solo estudiará los contenidos dados por el profesor sino que podrá mejorarlos a partir de las investigaciones que haga, lo cual no puede hacer con los textos de enseñanza. El computador le permite a través del programa PLEAC1, llegar a las lecciones y hacerle adiciones de los datos que el estudiante considere que pueden complementar y mejorar los temas dados por el instructor.

A través de la práctica del programa PLEAC1 se pudo observar como el usuario instructor no necesita conocer ningún lenguaje de programación - para utilizarlo en la creación de sus lecciones y cada profesor lo puede usar independientemente en su área, lo cual le confiere la característica de autor.

BIBLIOGRAFIA

ARIZMENDI, Octavio. La transformación educativa Nacional. Instituto Caro y Cuervo. Bogotá, 1969.

ATKINSON, Richard. Computer Asisted Instruction. Academic Press. Inc. New York, 1983.

BALLESTEROS, Fernando. Computadores en la educación. Universidad de los Andes. Bogotá, 1984.

BELTRAN, Carlos A. Sistemas de Enseñanza y Evaluación del Aprendizaje. Universidad del los Andres, Bogotá D.E., 1985.

BOEHM, Barry. Classics In Software Engineering.

BORK, Alfred. Computer and the future of education. Learning with computers. Bedford, Mass: Digital Press, 1981.

BORK, Alfred. The computers in teaching-widely Believed Myths. Learning or teaching with computer. Bedford, Mass. Digital Press, 1981.

DWNN, T. Rita. Procedimientos prácticos para individualizar la enseñanza. B.A. Argentina, Ed. Guadalupe, 1975.

FABER, Fair Uhlig. The ofice of the future. Vol. 1. New York. North Holland Publishing Col, 19881.

FINDLAY, William. Pascal. An introduction to Methodical programing. Rockville. Maryland, 1981.

FORO NACIONAL. A donde vá la Educación en Colombia. Bogotá, 1980.

FRANKLIN, Stephen. The role of personal computer systems in education. Bedford, Mass. Digital Press, 1981.

FRY, Edward B. Teaching machines and programmed instruction and Introduction. New York, McGraw Hill. 1963.

KELLER, Arthur M. Programación en Pascal. Stanford Univ. McGraw Hill México 1983.

LOW, Rodolfo. Compendio del Sistema educativo colombiano. Bogotá, 1971.

MARCOS, Juán. Instrucción asistida por Computador. Universidad Nacional Autónoma de México. México D.F., 1969.

O'SHEA, Tim. Self Jhon. Learning or teaching with computers. Englewood Cliffs. Prentice Hall Inc., 1983.

PIAGET, Jean. Educación e Instrucción. Buenos Aires. Edit. Proteo, 1970.

RATH ET AL, G.J. Anderson. N.C. Brainerd N.C. The IBM research center teaching Machine Project. New York. Jhon Wiley & Sons, Inc. 1959.

STARKWEATHER, Jhon A. A common Language for a variety of conversatinal programming needs. San Francisco, Universidad de California. Academic Press Inc., 1972.

STOLUROW, Lawrence. DAVIS, Daniel. Teaching Machines And Computer-based system. Teaching machines and Programmed Learning II. Washington d.c. National Educational Asociation of the United States, 1965.

WODD, Steve. Turbopascal versión 3.0. Osborne /Mc Graw-Hill, México 1986.

ANEXO 1. Especificaciones del Lenguaje de Comandos de PLEAC1

Especificaciones del Lenguaje de Comandos de PLEAC I.

```
<Comando>                 ::= .<Requerimiento>
<Requerimiento>           ::= <Espera> | <Frase_Incompleta> |
                              <Prueba_Corrección>| <Sonido> |
                              <Limpiar_Pantalla>| <Texto_Condicional> |
                              <FinalTexto_Condicional> |
                              <Punto_Interrupción> | <Asignación> |
                              <Comentario> | <Orden_Registro>
<Espera>                  ::= Et <Texto_Acotado> <Peso> {Vpeso}
                              {AccionET <Texto_Acotado>} <Final>
<Prueba_Corrección>       ::= PC <Codeva> <Formato> {AccionPC1} <Ayudas>
                              <Peso> {Vpeso} {AccionPC2} <Final>
<Frase_Incompleta>        ::= FI <Codeva> <Formato> {AccionFI1} <Ayudas>
                              <Peso> {Vpeso} {AccionFI2} <Final>
<Prueba_Asociación>       ::= PA <Codeva> <Pregunta> {AccionPA1}
                              <Respuesta> {AccionPA2} <Peso> {Vpeso}
                              {AccionPA3} <Final>
<Sonido>                  ::= PITO {Pitar} <Final>
<Limpiar_Pantalla>        ::= LP {Limpiar_Pantalla} <Final>
<Texto_Condicional>       ::= SI <Expresión>; <Final> {AccionSI}
<FinalTexto_Condicional>  ::= FINSI <Final>
<Punto_Interrupción>      ::= PI <Archivo> <Final> {AccionPI}
<Asignación>              ::= <Variable> := <Expresión>; {AccionAsig1}
                              <Final>
```

```
<Archivo>                  ::= <Texto_Acotado>

<Comentario>               ::= CO <Final>

<Orden_Registro>           ::= RG <Objeto_Registrar> [AccionRG] <Final>

<Objeto_Registrar>         ::= R/N

<Respuesta-Construida>     ::= RC <Codeva> <Formato> [AccionFI1] <Ayudas>
                               <Peso> [Vpeso] [AccionRC1] <Final>

<Codeva>                   ::= <Texto_Acotado>

<Peso>                     ::= <Texto-ACotado> [Validar valor entre 0 y
                               100]

<Texto-Acotado>            ::= <Separador> <Texto> <Separador>

<Expresión>                ::= <Expresión_Simple> | <Expresión-Simple>
                               <Relop> <Expresión_Simple> [AccionEXP]

<Relop>                    ::= < | > | <> | <= | >= | =

<Expresión_Simple>         ::= <Término> [<Operador_Aditivo> <Término>
                               [AccionEXP2]] | "Cadena"

<Operador_Aditivo>         ::= + | - | o

<Término>                  ::= <Signo> <Factor>[<Operador_Multiplicativo>
                              <Factor> [AccionTER2]] | "Cadena"

<Signo>                    ::= <Nulo> | + | -

<Operador_Multiplicativo>::= * | / | y

<Factor>                   ::= <Variable> [AccionFAC1] | <Constante>
                               [AccionFAC2] | (<Expresión>) | no<Factor>
                               [AccionFAC3]
```

<Respuesta> ::= <Texto_Acotado>

<Pregunta> ::= <Texto-Acotado>

<Formato> ::= <Texto_Acotado> <Respuesta> <Texto_Acotad>

<Ayuda> ::= <Texto_Acotado>

<Separador> ::= ! | " | @ | $ | % | & | / | (|) | = | ?
¡ | ¿ | + | * | { | } | [|] | < | > | ;
_ | .

ANEXO 2. Programa Registro

```
program registro;

Const
    ArchDesempeno = 'ENSEV001.DAT';
    ArchRespuestas= 'ENSEV002.DAT';
    ArchLecciones = 'ENSEV003.DAT';
    ArchPassWord  = 'ENSEV004.DAT';
    Max_Lec       = 75;



type
    Mensaje = string[100];
    Codigo  = string[6];
    Letras  = set of char;
    Cad255  = string[255];


    RegDesEst = record
              case integer of
              1:   (Pri_Est : integer);
              2:   (Nro_Est : Codigo;
                    Nom_Est : string[30];
                    Archivo : string[15];
                    Nro_Lec : integer;
                    Pra_Eva : integer;
                    Prx_Est : integer
                   );
              3:   (Cod_Eva : Codigo;
                    Puntaje : real;
                    Peso    : real;
                    Prx_Eva : integer
                   )
    end;

    reg_respuestas = record case integer of
    1: (            nro_est: codigo;
                    cod_eva: codigo;
                    peso_eva: real;
                    respuesta: string[255];
                    proximo: integer;
       );
    2: (            prim_eval: integer;
                    prim_libre: integer;
       );
    end;
```

```
    reg_lecciones = record
                  nom_lec: string[15];
    end;


Var
   RL      : array[0..99] of reg_lecciones;
   Terminar: boolean;
   Anterior: integer;
   RegActual: integer;
   AD: file of RegDesEst; {Archivo de estudiantes}
   AR: file of Reg_Respuestas;
   FClave: file of integer;
   BUFFER255: Cad255;
   Pw0,Pw3: Codigo;

procedure leerpw(var pw: codigo);
var i : integer;
begin
pw := '    ';
For i := 1 to 4 do
  begin
  read(kbd,pw[i]);
  write('*');
  end;
end;


Function NoPassWord(ORIG: Codigo): Boolean;
var i: integer;
begin
ClrScr;
Write('PROCEDIMIENTO CON ACCESO RESTRINGIDO');
GOTOXY(1,3);
WRITE('Cual es su clave de seguridad?     ');
i := 0;
Repeat
  GOTOXY(32,3);
  leerpw(pw3);
  i := i + 1;
until (pw0 = pw3) or (i = 3);
NoPassWord := pw0 <> pw3;
end;
```

```
procedure GetPw;
var v:boolean;
    i,j: integer;
begin
Assign(FClave,ArchPassWord);
{$i-}
Reset(FClave);
{$i+}
v := ioresult = 0;
pw0 := '    ';
if v then
  begin
  For i := 1 to 4 do
    begin
    read(FClave,j);
    Pw0[i] := chr(j);
    end;
  Close(Fclave);
  end;
end;


Function LeaReal(Columna,Linea: integer; R:Real):Real;
Var S:       String[20];
    REntrado:Real;
    Estado:  Integer;
begin
gotoxy(columna,linea);
write(R:6:2);
repeat
  gotoxy(Columna,Linea);
  readln(S);
  if S = '' then
    begin
    REntrado := R;
    Estado := 0;
    end
  else
    Val(S,REntrado,Estado);
until Estado = 0;
gotoxy(Columna,Linea);
write(REntrado:6:2);
LeaReal := REntrado;
end;
```

```
Function LeaEntero(Columna,Linea: integer; R:Integer):Integer;
Var S:       String[20];
    REntrado:Integer;
    Estado:  Integer;
begin
gotoxy(columna,linea);
write(R:6);
repeat
  gotoxy(Columna,Linea);
  readln(S);
  if S = '' then
    begin
    REntrado := R;
    Estado := 0;
    end
  else
    Val(S,REntrado,Estado);
until Estado = 0;
gotoxy(Columna,Linea);
write(REntrado:6);
LeaEntero := REntrado;
end;


Procedure LeaCad(Columna,Linea,Longitud: integer);
Var S1: Cad255;
    Estado:  Integer;
begin
gotoxy(columna,linea);
for Estado := 1 to Longitud do write('_');
gotoxy(columna,linea);
write(BUFFER255);
gotoxy(Columna,Linea);
readln(S1);
if S1 <> '' then
  BUFFER255 := s1;
if length(BUFFER255) > Longitud then
  BUFFER255 := copy(BUFFER255,Longitud+1,
                    Longitud-length(BUFFER255));
gotoxy(Columna,Linea);
write(' ',Longitud);
gotoxy(Columna,Linea);
write(BUFFER255);
 end;
```

```
Function Menu(s: Mensaje; o: letras): char;
var c:char;
begin
repeat
gotoxy(1,24);
write(' ':79);
gotoxy(1,24);
Write(s,' ');
read(c);
until (c in o);
gotoxy(1,24);
write(' ':79);
Menu := c;
end;

Function Existe(s: mensaje):boolean;
var f:file;
    ok:boolean;
begin
assign(f,s);
{$I-}
reset(f);
{$I+}
ok := (IOResult = 0);
if ok then close(f);
Existe := ok;
end;

procedure crear_desempeno;
var R: RegDesEst;
    f: file of RegDesEst;
begin
R.Pri_Est := 0;
Assign(f,ArchDesempeno);
rewrite(f);
write(f,R);
close(f);
end;
```

```
Function BusCod(CE: Codigo):integer;
var
   Encontro :  Boolean;
   Registro :  RegDesEst;
begin
Encontro := False;
Anterior := 0;
seek(AD,0);
Read(AD,Registro);
RegActual := Registro.Pri_Est;
While (RegActual <> 0) and (Not Encontro) do
  begin
  seek(AD,RegActual);
  Read(AD,Registro);
  if (Registro.Nro_Est)=CE) then
     Encontro := true
  else
     begin
     Anterior:=RegActual;
     RegActual := Registro.Prx_Est;
     end;
  end; ( while )
  If Encontro then if (registro.Nro_Est <> CE) then
                     BusCod := 0
                   else
                     BusCod := RegActual
  else
      BusCod := 0;
end;

Procedure LeaRegEst(VAR RE: RegDesEst);
Var REBuffer: RegDesEst;
begin
Gotoxy(5,10);
REBuffer := RE;
write('Nombre del estudiante: ');
gotoxy(35,10);
write(REBuffer.Nom_Est);
gotoxy(35,10);
Readln(REBuffer.Nom_Est);
If REBuffer.Nom_Est <> '' then
  RE.Nom_Est := REBuffer.Nom_Est;
end;
```

```
Procedure Mantenimiento;
var c          :char;
    CodEst     : Codigo;
    RAnterior,
    RE         : RegDesEst;
    Nreg       : Integer;
    Accion     : mensaje;
begin
IF NOPASSWORD('MTTO') then exit;
clrscr;
if not Existe(ArchDesempeno) then
  begin
  c := menu('El archivo de registro academico no existe.
            Lo creo? (Conteste S o N)',
     ['s','S','n','N']);
  if c in ['s','S'] then crear_desempeno
  else exit;
  end;
  Assign(AD,ArchDesempeno);
  reset(AD);

REPEAT
ClrScr;
gotoxy(5,5);
write('ENSEVAL: REGISTRO - Mantenimiento del archivo de
                           estudiantes');

  gotoxy(5,8);
  write('Codigo del estudiante: ');
  readln(CodEst);
  Nreg := BusCod(CodEst);
  If Nreg = 0 then
    begin
    RE.Nro_Est := CodEst;
    RE.archivo := '';
    RE.Nom_Est := '';
    RE.Pra_Eva := 0;
    RE.Nro_Lec := 0;
    RE.Prx_Est := 0;
    Nreg := filesize(AD);
```

```
    if (Anterior <> 0) then
      begin
      seek(AD,Anterior);
      read(AD,Ranterior);
      RE.Prx_Est := Ranterior.Prx_Est;
      Ranterior.Prx_Est := Nreg;
      seek(AD,Anterior);
      write(AD,Ranterior);
      end
    else (anterior es 0 y el Nreg debe quedar de primero en la cola)
      begin
      seek(AD,0);
      Read(AD,Ranterior);
      RE.Prx_Est := Ranterior.Pri_Est;
      Ranterior.Pri_Est := Nreg;
      seek(AD,0);
      write(AD,Ranterior);
      end;

    Accion := 'Adición      ';
    end
  else
    begin
    seek(AD,Nreg);
    read(AD,RE);
    Accion := 'Actualización';
    end;
  gotoxy(60,1);
  write(Accion);
  LeaRegEst(RE);
  Seek(AD,Nreg);
  write(AD,RE);
Until menu('MANTENIMIENTO: Continuar, Menu anterior',
                          ['M','m','C','c']) in ['m','M'];
close(AD); end;

Procedure Consulta;
var CodEst: codigo;
    Nreg:   integer;
    RE:     RegDesEst;
    RegEva: RegDesEst;
    SumaPeso,
    SumaNota,
    NotaTotal: Real;
    ContEval: integer;
    C:      Char;
    Salir:  Boolean;
```

```
Procedure RParcial;
Var  ActualEva:Integer;
     Salir: Boolean;
begin
gotoxy(1,14);
write(' ':160);
ActualEva := RE.Pra_Eva;
Salir := False;
While (Not Salir) AND (ActualEva <> 0) do
  begin
  seek(AD,ActualEva);
  Read(AD,RegEva);
  gotoxy(5,14);
  write('Nro.Eval');
  gotoxy(20,14);
  write('Nota');
  gotoxy(30,14);
  write('  Peso');
  gotoxy(5,15);
  Write(RegEva.Cod_Eva);
  gotoxy(20,15);
  write(RegEva.Puntaje:4:2);
  gotoxy(30,15);
  write(RegEva.Peso:6:2);
    case menu('CONSULTA EV.PARCIAL: Anterior eval. Menu anterior',
              ['a','m','A','M']) of
    'a','A':  ActualEva := RegEva.Prx_Eva;
    'm','M':  Salir:=true;
    end; {case}
  end; {while}
end;

Procedure RTotal;
Var  ActualEva:Integer;
begin
gotoxy(1,14);
write(' ':160);
ActualEva := RE.Pra_Eva;
gotoxy(20,14);
write('Nota');
gotoxy(30,14);
write('  Peso   (TOTAL SOBRE       EVALUACIONES)');
SumaPeso:=0;
SumaNota:=0;
ContEval:=0;
```

```
  While (ActualEva <> 0) do
    begin
    seek(AD,ActualEva);
    Read(AD,RegEva);
    SumaNota:=SumaNota+(RegEva.Puntaje*RegEva.Peso);
    SumaPeso:=SumaPeso+RegEva.Peso;
    ContEval:=ContEval+1;
    ActualEva:=RegEva.Prx_Eva;
    end; {while}
  if SumaPeso > 0 then
    begin
    NotaTotal := SumaNota/SumaPeso;
    gotoxy(53,14);
    Write(ContEval);
    gotoxy(20,15);
    write(NotaTotal:4:2);
    gotoxy(30,15);
    write(SumaPeso:6:2);
    end
  else
    begin
    gotoxy(1,14);
    write('No hay evaluaciones computables para
          el estudiante',' ':80);
    end;
  end;


begin
ClrScr;
if not Existe(ArchDesempeno) then
  begin
  gotoxy(1,21);
  writeln('El archivo de registro académico no existe.');
  writeln('Debe crearlo usando opción de MANTENIMIENTO');
  write('Presione ENTER para continuar');
  Readln;
  ClrScr;
  exit;
  end;
{ encontrar estudiante }
gotoxy(5,5);
write('ENSEVAL: REGISTRO - Consulta del archivo de estudiantes');
gotoxy(5,8);
write('Codigo del estudiante: ');
readln(CodEst);
```

```
Assign(AD,ArchDesempeno); reset(AD); Nreg := BusCod(CodEst);
{ si encuentra estudiante } if Nreg <> 0 then
  begin
  { mostrar datos basicos del estudiante y leccion actual }
  seek(AD,Nreg);
  read(AD,RE);
  Salir := False;
  While not Salir do
    begin
    Gotoxy(5,10);
    write('Nombre del estudiante: ');
    gotoxy(5,12);
    write('Leccion que cursa: ');
    gotoxy(35,10);
    write(RE.Nom_Est);
    gotoxy(35,12);
    write(RE.Nro_Lec,'(',RE.archivo,')');
    { mostrar menu R/egresar, Resultado P/arcial o T/otal }
    c:=Menu('CONSULTA: Menu anterior, resultado Parcial,
                       resultado Total',
       ['m','p','t','M','P','T']);
    { manejo de la seleccion hecha en ese menu }
      case c of
      'P','p': RParcial;
      'T','t': RTotal;
      'M','m': Salir := true;
      end;
    end;
  end
{ sino }
else
  { Mensaje de error y sale pa'pintura }
  begin
  gotoxy(1,22);
  writeln('El estudiante no existe en el registro Académico.');
  write('Presione ENTER para continuar');
  Readln;
  ClrScr;
  exit;
  end;
  close(AD);
end;
```

```
Procedure Editar;
var
  i,J,K,L,Q : integer;
  ya_existe,salir : Boolean;
  proc: text;
  curso,backcurso: INTEGER;
  opcion: char;
  Al : File of reg_lecciones;


    Procedure MostrarPlanLecciones;
    begin
    GOTOXY(1,1);
    WRITE('HAGA PLAN DE LECCIONES');
    J := MAX_LEC DIV 4 + 1;
    GOTOXY(1,3);
      FOR K := 0 TO J DO
      BEGIN
        FOR L := 0 TO 3 DO
        BEGIN
        if L <> 0 then I := K + (L * J + L)
        else I := K;
        IF I <= MAX_LEC THEN
          BEGIN
          WRITE(I:2,' - ');
          WRITE(RL[I].NOM_LEC);
          FOR Q := LENGTH(RL[I].NOM_LEC)+1 TO 15 DO WRITE(' ');
          END
        ELSE WRITE(' ':20);
        END;
      END;
    end;

    Procedure ObtenerCurso;
    begin
    GOTOXY(1,24);
    Write('Numero de leccion? ');
    Repeat
      curso := LeaEntero(20,24,curso);
    until curso in [0..Max_Lec];
    end;
```

```
    Procedure ObtenerNombreLeccion;
    begin
    Gotoxy(1,24);
    write(' ':79);
    Gotoxy(1,24);
    write('Nuevo nombre? ');
    BUFFER255 := RL[CURSO].NOM_LEC;
    LeaCad(15,24,15);
    RL[CURSO].NOM_LEC := BUFFER255;
    end;


begin
If noPASSWORD('EDIT') then exit;
curso := 0;
ClrScr;
  if not existe (ArchLecciones) then
  begin
  Assign(Al,ArchLecciones);
  rewrite(Al);
    for i:=0 to Max_lec do
      begin
      Rl[i].Nom_Lec := '';
      write(Al,RL[i]);
      end;
  close(AL);
  end;
Assign(al,ArchLecciones);
Reset(AL);
for i := 0 to Max_Lec do read(Al,RL[i]);
close(AL);
salir := false;
REPEAT
  MostrarPlanLecciones;
  opcion := Menu('OPERACION A REALIZAR: Editar, Suprimir,
                      Insertar, Cambiar, Menu anterior',
       ['M','m','E','e','C','c','S','s','I','i']);
  case opcion of
       'm','M': Salir := true;
       'e','E':Begin
               ObtenerCurso;
               if RL[Curso].Nom_Lec = ''
               then ObtenerNombreLeccion;
               Assign(proc,'P.BAT');
               rewrite(proc);
               Writeln(proc,'ECHO OFF');
               Writeln(proc,'CLS');
               Writeln(proc,'ED ',RL[Curso].Nom_Lec);
               Writeln(proc,'CLS');
               Writeln(proc,'Q');
               Close(proc);
               Close(Al);
               Terminar := True;
```

```
        'C','c':Begin
                ObtenerCurso;
                ObtenerNombreLeccion;
                end;
        'S','s':Begin
                ObtenerCurso;
                While Curso <= Max_Lec do
                  begin
                  RL[Curso] := RL[Curso+1];
                  curso := curso + 1;
                  end;
                RL[Max_Lec].Nom_Lec := '';
                End;
        'I','i':Begin
                if RL[Max_Lec].Nom_Lec <> '' then
                  begin
                  gotoxy(1,24);
                  write(' ':80);
                  gotoxy(1,24);
                  write('No puede insertar ni adicionar lecciones');
                  end
                else
                  begin
                  ObtenerCurso;
                  backcurso := Max_lec;
                  while backcurso > curso do
                    begin
                    RL[backcurso] := RL[backcurso-1];
                    backcurso := backcurso - 1;
                    end;
                  RL[curso].Nom_lec := '';
                  end;
                End;
  end;
UNTIL TERMINAR or SALIR;
Assign(al,ArchLecciones);
Reset(AL);
for i := 0 to Max_Lec do write(Al,RL[i]);
close(AL);
end;
```

```
Procedure Pw;
var Pw1,Pw2: codigo;
    i,j: integer;


begin
ClrScr;
If NoPassWord('PW') then exit;
ClrScr;
Repeat
      Gotoxy(1,1);
      Writeln('Cambio de PassWord');
      Gotoxy(1,3);
      Write('Escriba su nueva clave:      ');
      Gotoxy(26,3);
      leerpw(pw1);
      Gotoxy(1,5);
      Write('Confirme su nueva clave:     ');
      gotoxy(26,5);
      leerpw(pw2);
Until Pw1 = Pw2;
PWO := Pw1;
Assign(FClave,ArchPassWord);
Rewrite(FClave);
For i := 1 to 4 do
  begin
  j := Ord(PwO[i]);
  write(FClave,J);
  end;
Close(Fclave);
end;

Procedure RegProg;
Var
   RespActual: integer;
   LibreActual: integer;
   Resp: Reg_Respuestas;
   Terminar: Boolean;
```

```
Procedure Calificar;
Var Nreg:integer;
    EvalExiste: boolean;
    ActualEva: integer;
    RegEst: RegDesEst;
    RegEva: RegDesEst;
    NRegEva: integer;
begin
{ ya conoce el codigo del estudiante: localizarlo en
  archivo desempeño }
if (not Existe(ArchDesempeno)) then
  begin
  gotoxy(1,21);
  writeln('El archivo de registro académico no existe.');
  writeln('Debe crearlo usando opción de MANTENIMIENTO');
  write('Presione ENTER para continuar');
  Readln;
  ClrScr;
  exit;
  end;
  assign(AD,ArchDesempeno);
  reset(AD);
  Nreg := BusCod(Resp.Nro_Est);
  If Nreg = 0 then
    begin
    gotoxy(1,21);
    writeln('ERROR.- Estudiante # ',Resp.Nro_Est,' no inscrito.');
    writeln('        No se admite calificacion.');
    Readln;
    ClrScr;
    Exit;
    end;
{ recorrer cola de evaluaciones estudiante, para evitar
  duplicacion de la evaluacion.
}
Seek(AD,Nreg);
Read(AD,RegEst);
EvalExiste := false;
ActualEva := RegEst.Pra_Eva;
While (ActualEva <> 0) and (Not EvalExiste) do
  begin
  seek(AD,ActualEva);
  Read(AD,RegEva);
  EvalExiste := (RegEva.cod_eva = Resp.cod_eva);
  if not EvalExiste then
    ActualEva := RegEva.Prx_Eva;
  end;
{ si la evaluacion existe debe presentar la nota y
  pedir la nueva nota
}
```

```
    if not EvalExiste then
     begin
      (* adicionar evaluacion con peso dado en texto del curso *)
      NRegEva := filesize(AD);
      seek(AD,NRegEva);
      RegEva.Cod_Eva := Resp.Cod_Eva;
      RegEva.Puntaje := 0;
      RegEva.Peso := Resp.Peso_Eva;
      RegEva.Prx_Eva := RegEst.Pra_Eva;
      RegEst.Pra_Eva := NRegEva;
      Seek(AD,NRegEva);
      Write(AD,RegEva);
      Seek(AD,NReg);
      Write(AD,RegEst);
      end
    else
      begin
      NRegEva := ActualEva;
      Seek(AD,NRegEva);
      Read(AD,RegEva);
      end;
    gotoxy(1,10);
    Writeln('Puntaje (0.00 a 5.00): ');
    RegEva.Puntaje := LeaReal(30,10,RegEva.Puntaje);
    gotoxy(1,12);
    writeln('Peso (0.00 a 100.00):  ');
    RegEva.Peso := LeaReal(30,12,RegEva.Peso);
    Seek(AD,NRegEva);
    write(AD,RegEva);
    close(AD);
    end;

begin
If NoPASSWORD('RGPG') then exit;
if not Existe(ArchRespuestas) then
  begin
  gotoxy(1,21);
  writeln('El archivo de respuestas no existe.');
  writeln('No hay respuestas pendientes.');
  write('Presione ENTER para continuar');
  Readln;
  ClrScr;
  exit;
  end;
```

```
assign(AR,ArchRespuestas); reset(AR); seek(AR,0);
read(AR,Resp); RespActual := Resp.Prim_Eval; LibreActual :=
Resp.Prim_Libre; Terminar := False; While (RespActual <> 0)
and (Not Terminar) do
  begin
  seek(AR,RespActual);
  read(AR,Resp);
  ClrScr;
  Gotoxy(1,1);
  Writeln('Estudiante: ',Resp.Nro_est);
  Writeln('Codigo evaluacion: ',Resp.Cod_Eva);
  Writeln;
  Writeln('Respuesta');
  Writeln(Resp.Respuesta);
  case Menu('REGISTRO DE PROGRESO: Menu Anterior,
                Calificar, Proxima evaluacion ',
       ['M','m','C','c','P','p']) of
  'M','m': Terminar := True;
  'P','p': RespActual := Resp.proximo;
  'C','c': Calificar;
  end; (case)
  end;
  Close(AR);
end;


begin
clrscr;
GetPw;
Terminar := false;
while not Terminar do
  case menu('REGISTRO: Mantenimiento, Consulta,
            Registro progreso, Editar, Terminar, Pw',
           ['P','p','E','e','M','m','C','c','R','r','T','t']) of
  'M','m': Mantenimiento;
  'C','c': Consulta;
  'R','r': RegProg;
  'T','t': Terminar:=true;
  'E','e': Editar;
  'P','p': Pw;
  end;
end.
```

ANEXO 3. Programa Presenta

```
Program presenta;

Const
     MaxAsociaciones = 29;
     ArchDesempeno   = 'ENSEV001.DAT';
     ArchRespuestas  = 'ENSEV002.DAT';
     ArchNumeriones  = 'ENSEV003.DAT';
     Max_Lec         = 75;
     MAX_UNGET       = 12;


type
    nombrearchivo= string[20];
    Codigo = string[6];
    CAD4   = string[4];
    CAD12  = string[12];
    Cad255 = STRING[255];
    letras = set of char;
    TablaParaPA = Array [0..MaxAsociaciones] of Cad4;


    TTOKEN = (Et,                 { Esperar Tecla }
              Rc,                 { Respuesta Construida }
              Pc,                 { Prueba de Correccion }
              Fi,                 { Frase Incompleta }
              Pa,                 { Prueba de Asociacion }
              Pito,               { Efecto Sonoro }
              Lp,                 { Limpiar Pantalla }
              Si,                 { Condicional }
              FinSi,              { Fin Condicional }
              Pi,                 { Punto de Interrupcion }
              Asignacion,         { := }
              Co,                 { Comentario }
              Rg,                 { Registrar ... }
              R,                  { ... Respuesta }
              N,                  { ... Nota }
              MenorQue,           { < }
              MayorQue,           { > }
              Diferente,          { <> }
              MenorOIgual,        { <= }
              MayorOIgual,        { >= }
              Igual,              { = }
              Comilla,            { " }
              Mas,                { + }
              Menos,              { - }
              OLogica,            { O }
              Multiplicar,        { * }
              Dividir,            { / }
              YLogico,            { Y }
              ParentIzq,          { ( }
              ParentDer,          { ) }
              NoLogico,           { NO }
              Variable,           { Nombre de variable }
              CteNumerica,        { Cadena de digitos }
```

```
RTOKEN = RECORD
       VALOR:STRING[12];
       CODIGO:TTOKEN;
END;

RegDesEst = record
          case integer of
          1:   (Pri_Est : integer);
          2:   (Nro_Est : Codigo;
                Nom_Est : string[30];
                Archivo : string[15];
                Nro_Lec : integer;
                Pra_Eva : integer;
                Prx_Est : integer
               );
          3:   (Cod_Eva : Codigo;
                Puntaje : real;
                Peso    : real;
                Prx_Eva : integer
               )
end;

reg_respuestas = record case integer of
1: (           nro_est: codigo;
               cod_eva: codigo;
               peso_eva: real;
               respuesta: string[255];
               proximo: integer;
   );
2: (           prim_eval: integer;
               prim_libre: integer;
   );
end;


reg_lecciones = record
              nom_lec: string[15];
end;



TIPOS = (NUMERICO, CARACTER);

PTVariable = ^TVariable;

TVariable = RECORD
          ProxVar: PTVariable;
          TipoDeVar: TIPOS;
          NombreVar:Cad12;
          CASE INTEGER OF
               NUMERICO: (Valor: real);
               CARACTER: (Cadena: ^Cad255);
END;
```

```
var
   PO_J, PRE_J: PTVariable;
   curso: text;
   LeccionActual : string[15];
   SioNo: char; { buffer para "ungetchar(c: char)" }

   lastchar: array [1..MAX_UNGET] of char;
   nextunget,
   nextget:  integer;

   TablaToken: Array [TTOKEN] of String[12];
   LastToken : RToken;

   PrimVar: PTVariable;

   LineaCursorRespuesta, ColumnaCursorRespuesta,
   LongitudRespuesta: Integer;
   LineaCursorSeguir, ColumnaCursorSeguir : Integer;

   BUFFER255: Cad255;
   AbandonarEnPI: Boolean;

   RespOKDePA: TablaParaPA;

   RL      : array[0..Max_Lec] of reg_lecciones;
   NroRegEst : integer;
   Anterior: integer;
   RegActual: integer;
   AD: file of RegDesEst; (Archivo de estudiantes)
   RegEst: RegDesEst;
   AR: file of Reg_Respuestas;
   Al: File of reg_lecciones;
   RR: Reg_Respuestas;


Procedure PExpresion(var ValExpr: TVariable);
Forward;


function LinCur: integer;
var Linea  : char absolute 0:1105;
begin
LinCur := Ord(Linea)+1;
end;

function ColCur: integer;
var Columna: char absolute 0:1104;
begin
ColCur := Ord(Columna)+1;
end;
```

```
Function BusCod(CE: Codigo):integer;
var
   Encontro :  Boolean;
   Registro :  RegDesEst;
begin
Encontro := False;
Anterior := 0;
seek(AD,0);
Read(AD,Registro);
RegActual := Registro.Pri_Est;
While (RegActual <> 0) and (Not Encontro) do
  begin
  seek(AD,RegActual);
  Read(AD,Registro);
  if (Registro.Nro_Est>=CE) then
     Encontro := true
  else
     begin
     Anterior:=RegActual;
     RegActual := Registro.Prx_Est
     end;
  end; ( while )
  If Encontro then if (registro.Nro_Est <> CE) then
                     BusCod := 0
                   else
                     BusCod := RegActual
  else
      BusCod := 0;
end;
```

```
Procedure IniciarTablaToken;
begin
TablaToken[Et] := 'ET';
TablaToken[Rc] := 'RC';
TablaToken[Pc] := 'PC';
TablaToken[Fi] := 'FI';
TablaToken[Pa] := 'PA';
TablaToken[Pito] := 'PITO';
TablaToken[Lp] := 'LP';
TablaToken[Si] := 'SI';
TablaToken[FinSi] := 'FINSI';
TablaToken[Pi] := 'PI';
TablaToken[Asignacion] := ':=';
TablaToken[Co] := 'CO';
TablaToken[Rg] := 'RG';
TablaToken[R] := 'R';
TablaToken[N] := 'N';
TablaToken[MenorQue] := '<';
TablaToken[MayorQue] := '>';
TablaToken[Diferente] := '<>';
TablaToken[MenorOIgual] := '<=';
TablaToken[MayorOIgual] := '>=';
TablaToken[Igual] := '=';
TablaToken[Comilla] := '"';
TablaToken[Mas] := '+';
TablaToken[Menos] := '-';
TablaToken[OLogica] := 'O';
TablaToken[Multiplicar] := '*';
TablaToken[Dividir] := '/';
TablaToken[YLogico] := 'Y';
TablaToken[ParentIzq] := '(';
TablaToken[ParentDer] := ')';
TablaToken[NoLogico] := 'NO';
TablaToken[Variable] := '';
TablaToken[CteNumerica] := '';
TablaToken[NoToken] := '';
end;

Procedure ErrorFatal(msg: Cad255);
begin
writeln;
writeln;
Write(msg);
writeln;
writeln;
ReadLn;
end;
```

```
Procedure Beep;
begin
Write(char(7));
end;

Function LeaReal(Columna,Linea: integer; R:Real):Real;
Var S:       String[20];
    REntrado:Real;
    Estado:  Integer;
begin
gotoxy(columna,linea);
write(R:6:2);
repeat
  gotoxy(Columna,Linea);
  readln(S);
  if S = '' then
    begin
    REntrado := R;
    Estado := 0;
    end
  else
    Val(S,REntrado,Estado);
until Estado = 0;
gotoxy(Columna,Linea);
write(REntrado:6:2);
LeaReal := REntrado;
end;

Function LeaEntero(Columna,Linea: integer;
                   R:Integer):Integer;
Var S:       String[20];
    REntrado:Integer;
    Estado:  Integer;
begin
gotoxy(columna,linea);
write(R:6);
repeat
  gotoxy(Columna,Linea);
  readln(S);
  if S = '' then
    begin
    REntrado := R;
    Estado := 0;
    end
  else
    Val(S,REntrado,Estado);
until Estado = 0;
gotoxy(Columna,Linea);
write(REntrado:6);
LeaEntero := REntrado;
end;
```

```
Procedure LeaCad(Columna,Linea,Longitud: integer);
Var S1: Cad255;
    Estado:  Integer;
begin
gotoxy(columna,linea);
for Estado := 1 to Longitud do write('_');
gotoxy(columna,linea);
write(BUFFER255);
gotoxy(Columna,Linea);
readln(S1);
if S1 <> '' then
  BUFFER255 := s1;
if length(BUFFER255) > Longitud then
  BUFFER255 :=
copy(BUFFER255,Longitud+1,Longitud-length(BUFFER255));
gotoxy(Columna,Linea);write(' ',Longitud);
gotoxy(Columna,Linea); write(BUFFER255); end;


Function Menu(s: Cad255; o: letras): char;
var c:char;
begin
repeat
gotoxy(1,24);
write(' ':79);
gotoxy(1,24);
Write(s,' ');
read(c);
until (c in o);
gotoxy(1,24);
write(' ':79);
Menu := c;
end;

Function Existe(s: cad255):boolean;
var f:file;
    ok:boolean;
begin
assign(f,s);
{$I-}
reset(f);
{$I+}
ok := (IOResult = 0);
if ok then close(f);
Existe := ok;
end;
```

```
Procedure UnGetChar(c: char);
begin
lastchar[nextunget] := c;
nextunget := nextunget + 1;
end;

function getchar: char;
var c:char; (buffer)
begin
if nextget < nextunget then { hay lastchar para re-leer }
  begin
  c := lastchar[nextget];
  lastchar[nextget] := char(0);
  nextget := nextget + 1;
  if nextget=nextunget then
    begin
    nextget := 1;
    nextunget := 1;
    end;
  end
else
  begin
  read(curso,c);
  end;
getchar := c;
end;

Procedure DefinirVariable(Tipo:tipos;Nombre:Cad12);
var p: PTVariable;
begin
new(p);
p^.ProxVar := PrimVar;
PrimVar := p;
p^.TipoDeVar := Tipo;
p^.NombreVar := Nombre;
Case Tipo of
  Numerico: P^.Valor := 0;
  Caracter:
    Begin
    New(P^.Cadena);
    P^.Cadena^ := '';
    End;
  end; { case }
end;
```

```
Procedure UbicarVariable
(Var Anterior,VarPointer: PTVariable;Nom:Cad12);
 var Encontro: Boolean;
 Begin VarPointer := PrimVar;
 Anterior := Nil;
 Encontro := false;
 While (VarPointer <> nil) and (Not Encontro) do
  begin
  if NOM = VarPointer^.NombreVar then
    Encontro := True
  Else
    begin
    Anterior := VarPointer;
    VarPointer := VarPointer^.ProxVar;
    end;
  end; { while }
End;

Procedure StoreNum(Vlr: Real; Nom: Cad12);
var PreP,P: PTVariable;
begin
UbicarVariable(PreP,P,Nom);
If P = Nil then
  begin
  DefinirVariable(Numerico,Nom);
  UbicarVariable(Prep,P,Nom);
  If P = Nil then
    begin
    ErrorFatal('No pude crear variable '+Nom);
    Exit;
    end;
  end;
If p^.TipoDeVar <> Numerico then
  begin
  ErrorFatal('Inconsistencia en tipos');
  exit;
  end;
P^.NombreVar := Nom;
P^.Valor := Vlr;
end;
```

```
Procedure StoreCad(Vlr: Cad255; Nom: Cad12);
var PreP,P: PTVariable;
begin
UbicarVariable(PreP,P,Nom);
If P = Nil then
  begin
  DefinirVariable(Caracter,Nom);
  UbicarVariable(Prep,P,Nom);
  If P = Nil then
    begin
    ErrorFatal('No pude crear variable '+Nom);
    Exit;
    end;
  end;
If p^.TipoDeVar <> Caracter then
  begin
  ErrorFatal('Inconsistencia en tipos');
  exit;
  end;
P^.NombreVar := Nom;
P^.Cadena^ := Vlr;
end;

Function EsEspacio(c:char):boolean;
begin
  case c of
     #9,#10,#13,#32: EsEspacio := true;
     else EsEspacio := false;
  end;
end;

Procedure UnGetToken(token: RToken);
begin
LastToken := token;
end;

Procedure GetToken(Var Token: RToken);
var c: char;
    cc: string[1];
    CadTok:string[12];

    Procedure Tokenizar;
    var i: TToken;
    begin
    for i := ET to NoLogico do
      if TablaToken[i] = CadTok then
        begin
        Token.Valor := CadTok;
        Token.Codigo := i;
        end;
    end;
```

```
begin
if LastToken.Codigo = NoToken then
  begin
  CadTok := '';
  Token.Valor := '';
  Token.Codigo := NoToken;
  c := UpCase(getchar);
  while (EsEspacio(c)) do c := UpCase(getchar);
    case c of
    '=','"','+','-','*','/','(',')':
        Begin
        CadTok := ' ';
        CadTok[1] := c;
        Tokenizar;
        end;
    ':':
        Begin
        CadTok := ' ';
        CadTok[1] := c;
        c := UpCase(getchar);
        cc := ' ';
        cc[1] := c;
        CadTok := CadTok + cc;
        Tokenizar;
        end;
    '<':
        Begin
        CadTok := ' ';
        CadTok[1] := c;
        c := UpCase(getchar);
        if c in ['>','='] then
              begin
              cc := ' ';
              cc[1] := c;
              CadTok := CadTok + cc;
              end
        else
              UnGetChar(c);
        Tokenizar;
        end;
```

```
'>':
    Begin
    CadTok := ' ';
    CadTok[1] := c;
    c := UpCase(getchar);
    if c in ['='] then
          begin
          cc := ' ';
          cc[1] := c;
          CadTok := CadTok + cc;
          end
    else
          UnGetChar(c);
    Tokenizar;
    end;
'A'..'Z':
    Begin
    CadTok := ' ';
    CadTok[1] := c;
    c := UpCase(getchar);
    while c in ['A'..'Z','0'..'9'] do
      begin
      cc := ' ';
      cc[1] := c;
      CadTok := CadTok + cc;
      c := UpCase(getchar);
      end;
    UnGetChar(c);
    Tokenizar;
    if Token.Codigo = NoToken then
      begin
      Token.Valor := CadTok;
      Token.Codigo := Variable;
      end;
    end;
```

```
    '0'..'9':
        Begin
        CadTok := ' ';
        CadTok[1] := c;
        c := UpCase(getchar);
        while c in ['0'..'9','.'] do
          begin
          cc := ' ';
          cc[1] := c;
          CadTok := CadTok + cc;
          c := UpCase(getchar);
          end;
        UnGetChar(c);
        Tokenizar;
        if Token.Codigo = NoToken then
          begin
          Token.Valor := CadTok;
          Token.Codigo := CteNumerica;
          end;
        end;
    end; { case }
  end
else
  begin
  Token := LastToken;
  LastToken.Codigo := NoToken;
  LastToken.Valor := '';
  end; (else)
end;

procedure abrircurso(s:nombrearchivo);
begin
ClrScr;
assign(curso,s);
reset(curso);
end;

Procedure Pfinal;
var I:integer;
    c:char;
Begin
i := 0;
while (not EOF(curso)) and (i<2) do
  begin
  c := getchar;
  if c in [#10,#13] then
    begin
    i := i + 1;
    if i>2 then i:= i-2;
    end;
  End;
end;
```

```
Procedure PComentario;
begin
PFinal;
end;

Procedure PSonido;
begin
BEEP;
Pfinal;
end;

Procedure PTextoAcotado(var S:Cad255);
var c:char;
    separador: char;
    cc: string[1];
begin
s := '';
c := getchar;
while EsEspacio(c) do c:=getchar;
if c in ['!','#','$','&','(',')','*','+',',','-','.','/',
         ':',';','<','=','>','?','@','[',']','^','_','{',
         '}','\','"','%'] then
         separador := c
else
  begin
  cc := ' ';
  cc[1] := c;

  ErrorFatal('Texto acotado invalido'+'['+cc+']');
  exit;
  end;
c := getchar;
while c <> separador do
  begin
  cc := ' ';
  cc[1] := c;
  S := S+cc;
  c := getchar;
  end;
end;
```

```
Procedure PCodEva(var CodEva: Codigo);
var s:cad255;
    p,prep: ptvariable;
begin
PTextoAcotado(s);
If (length(s)<1) or (length(s)>6) then
  begin
  errorFatal('Codigo de evaluacion invalido: '+s);
  exit;
  end;
  UBICARVARIABLE(PREP,P,'CODEVA');
  if P <> nil then
    P^.Cadena^ := s
  else
    ErrorFatal('Error de programa - No existe variable
                ambiental CODEVA');
 end;

Procedure AccionET(S: cad255);
var c: char;
    i: integer;
    NroIntentos: integer;
    ok: boolean;
begin
NroIntentos := 0;
repeat
  NroIntentos := NroIntentos+1;
  Read(Kbd,c);
  ok := false;
  for i:= 1 to length(S) do
    ok := ok or (s[i] = c);
until ok;
StoreNum(NroIntentos,'NOTA');
end;

Procedure PPeso(var N:real);
var TextoAcotado: cad255;
    Peso: Real;
    Estado: integer;
begin
PTextoAcotado(TextoAcotado);
VAL(TextoAcotado,Peso,Estado);
If (Estado = 0) and (Peso > 0) and (Peso < 100) then N := Peso
else
  Begin
  N := 0;
  ErrorFatal('Peso de evaluacion invalido');
  end;
end;
```

```
Procedure VPeso(peso:real);
begin
StoreNum(peso,'PESO');
end;

Procedure PEspereTecla;
var TextoAcotado: Cad255;
    Peso: real;
begin
PTextoAcotado(TextoAcotado);
PPeso(peso);
Vpeso(peso);
AccionET(TextoAcotado);
Pfinal;
end;

Procedure PRespuesta(Var Respuesta: Cad255);
begin
PTextoAcotado(Respuesta);
end;

Procedure PPregunta(Var Pregunta: Cad255);
begin
PTextoAcotado(Pregunta);
end;

Procedure PAyuda(Var Ayuda: Cad255);
begin
PTextoAcotado(Ayuda);
end;

Procedure PFormato(Var Texto1,Texto2,Texto3: cad255);
begin
PTextoAcotado(Texto1);
PRespuesta(Texto2);
PTextoAcotado(Texto3);
end;
```

```
Procedure AccionFI1(Texto1,Texto2,Texto3: Cad255);
var i:integer;
begin
Write(Texto1);
LongitudRespuesta := length(Texto2)+length(Texto2) mod 7;
LineaCursorRespuesta := LinCur;
ColumnaCursorRespuesta := ColCur;
i := LongitudRespuesta;
While i >= 0 do
  begin
  write('_');
  i := i - 1;
  end;
Write(Texto3);
Writeln;
LineaCursorSeguir := LinCur;
ColumnaCursorSeguir := ColCur;
end;

Procedure Depurar(var s:cad255);
Var i : integer;
begin
For i := 1 to length(s) do
  if (s[i] < ' ') or (s[i] > '~') then s[i]:=' ';
end;

Procedure AYUDAR(s: cad255);
begin
GotoXY(ColumnaCursorSeguir,LineaCursorSeguir);
Writeln;
Writeln('AYUDA:');
Writeln;
Depurar(s);
Writeln(s);
LineaCursorSeguir := LinCur;
ColumnaCursorSeguir := ColCur;
end;
```

```
Procedure AccionFI2(Respuesta,ayuda: Cad255);
var OK: Boolean;
    NroIntentos : integer;
begin
NroIntentos := 0;
BUFFER255 := '';
Repeat
  NroIntentos := NroIntentos + 1;
  if NroIntentos = 2 then AYUDAR(ayuda);
  LeaCad(ColumnaCursorRespuesta,LineaCursorRespuesta,
        LongitudRespuesta);
  ok := (BUFFER255 = Respuesta);
until OK;
StoreNum(NroIntentos,'NOTA');
GotoXY(ColumnaCursorSeguir,LineaCursorSeguir);
end;

Procedure PFraseIncompleta;
var Texto1,RespuestaOK,Texto2,ayuda: cad255;
    peso: real;
    CodEva: Codigo;
begin
Writeln;
Writeln('Presione ENTER para continuar');
AccionET(#13);
ClrScr;
PCodEva(CodEva);
PFormato(Texto1,RespuestaOK,Texto2);
AccionFI1(Texto1,RespuestaOK,Texto2);
PAyuda(ayuda);
PPeso(peso);
Vpeso(peso);
AccionFI2(RespuestaOK,Ayuda);
Pfinal;
end;
```

```
Procedure AccionRC1(RespuestaOK,Ayuda: Cad255);
var NroIntentos: integer;
    ok:          boolean;

    Function LR: integer;
    begin
    if LongitudRespuesta = 0 then LR := 255
    else LR := LongitudRespuesta;
    end;

begin
NroIntentos := 0;
BUFFER255 := '';
Repeat
  NroIntentos := NroIntentos + 1;
  if NroIntentos = 2 then AYUDAR(ayuda);
  LeaCad(ColumnaCursorRespuesta,LineaCursorRespuesta,LR);
  if LongitudRespuesta > 0 then
    ok := (BUFFER255 = RespuestaOK)
  else
    begin
    ok := true;
    StoreCad(BUFFER255,'RESPUESTA');
    end;
until OK;
StoreNum(NroIntentos,'NOTA');
GotoXY(ColumnaCursorSeguir,LineaCursorSeguir);
end;

Procedure PRespuestaConstruida;
var Texto1,RespuestaOK,Texto2,ayuda: cad255;
    peso: real;
    CodEva: Codigo;
begin
Writeln;
Writeln('Presione ENTER para continuar');
AccionET(#13);
ClrScr;
PCodEva(CodEva);
PFormato(Texto1,RespuestaOK,Texto2);
AccionFI1(Texto1,RespuestaOK,Texto2);
PAyuda(ayuda);
PPeso(peso);
Vpeso(peso);
AccionRC1(RespuestaOK,Ayuda);
Pfinal;
end;
```

```
Procedure AccionPC1(TextoAcotado: cad255);
begin
Write(TextoAcotado);
LineaCursorRespuesta := LinCur;
ColumnaCursorRespuesta := ColCur;
end;

Procedure AccionPC2(RespuestaNoOK,RespuestaOK,ayuda: Cad255);
var OK: Boolean;
    NroIntentos : integer;
begin
NroIntentos := 0;
BUFFER255 := RespuestaNoOK;
LongitudRespuesta := length(RespuestaOK)+Length(RespuestaOK) Mod 7;
Repeat
  NroIntentos := NroIntentos + 1;
  if NroIntentos = 2 then AYUDAR(ayuda);
  LeaCad(ColumnaCursorRespuesta,LineaCursorRespuesta,
         LongitudRespuesta);
  ok := (BUFFER255 = RespuestaOK);
Until OK;
StoreNum(NroIntentos,'NOTA');
GotoXY(ColumnaCursorSeguir,LineaCursorSeguir);
end;

Procedure PPruebaCorreccion;
var Texto1,RespuestaNoOK,RespuestaOK,ayuda: cad255;
    peso: real;
    CodEva: Codigo;
begin
Writeln;
Writeln('Presione ENTER para continuar');
AccionET(#13);
ClrScr;
PCodEva(CodEva);
PFormato(Texto1,RespuestaNoOK,RespuestaOK);
AccionPC1(Texto1);
PAyuda(ayuda);
PPeso(peso);
Vpeso(peso);
AccionPC2(RespuestaNoOK,RespuestaOK,Ayuda);
Pfinal;
end;
```

```
Procedure OrdenarAsociaciones(var VectorPA: TablaParaPA;
                              n: integer);
 var i,j: integer;
  Procedure swap;
  var t: cad4;
  begin
  t := VectorPA[i];
  VectorPA[i] := VectorPA[j];
  VectorPA[j] := t;
  end;
begin
for i := 0 to n-1 do
  for j := i+1 to n do
    if VectorPA[i] >= VectorPA[j] then swap;
end;

Procedure AccionPA1(pregunta: cad255);
begin
Write(pregunta);
LineaCursorRespuesta := LinCur;
ColumnaCursorRespuesta := ColCur;
end;

Procedure AccionPA2(R255:cad255;Var
                    RespuestasDePA:TablaParaPA;
                    Var j: integer; Profe: boolean);

var i,LongDeR: integer;
    cc: string[1];
    subsecuencia: cad255;
    R: Array [1..256] of char;
begin

for i := 1 to length(R255) do R[i] := R255[i];
LongDeR := length(r255)+1;
r[LongDeR]:=',';
```

```
{ Separar subcadenas en Array of Cad4 global.}
for i := 0 to MaxAsociaciones do { iniciar vector de
                                   respuestas correctas }
  RespuestasDePA[i] := '';
for i := 1 to LongDeR do R[i] := UpCase(R[i]);
j := 0;
subsecuencia := '';
for i := 1 to LongDeR do
  if R[i] in ['A'..'Z','0'..'9'] then
    begin
    cc := ' ';
    cc[1] := R[i];
    subsecuencia := subsecuencia + cc;
    end
  else if R[i] = ',' then
        begin
        if length(subsecuencia) > 4 then
          begin
          ErrorFatal('Una de las asociaciones para Prueba
                     de Asociacion Excede 4 caracteres');
          exit;
          end;
        RespuestasDePA[j] := Subsecuencia;
        Subsecuencia := '';
        j:=j+1;
        if j > MaxAsociaciones then
          begin
          ErrorFatal('Ha excedido la capacidad de la tabla
                 de relaciones para Prueba de asociacion');
          exit;
          end;
        end
      else Begin
           ErrorFatal('Caracter invalido en respuesta de
                      Prueba de Asociacion');
           Exit;
           end;
{ Ordenarlos Ascendentemente. }
j := j-1;
if (j<=0) and (Profe) then
  begin
  errorfatal('No se admite Prueba de Asociacion sin
              respuestas del profesor');
  exit;
  end;
If RespuestasDePA[j] = '' then
  begin
  ErrorFatal('Respuesta invalida para Prueba de Asociacion');
  exit;
  end;
OrdenarAsociaciones(RespuestasDePA,j);
end;
```

```
Procedure AccionPA3(NroAsocs: integer);
var Aciertos: TablaParaPA;
    i,l,c: integer;
    NroAsEst: integer;
    NroIntentos: integer;
    NroAciertos: integer;
    RespEsAPA: TablaParaPA;

    Procedure CHEQUEO;
    var I,J: integer;

      procedure compress;
      var k:integer;
      begin
      k := i;
      while k < NroAsocs do
        begin
        RespOKDePA[k] := RespOKDePA[k+1];
        k := k + 1;
        end;
      RespOKDePA[NroAsocs] := '';
      NroAsocs := NroAsocs - 1;
      end;

    begin
    i := 0;
    j := 0;
    while (i<=NroAsocs) and (j<=NroAsEst) do
      begin
      if RespEsAPA[j] = RespOKDePA[i] then
        begin
        NroAciertos := NroAciertos + 1;
        AciertosCNroAciertos] := RespOKDePA[i];
        Compress;
        j := j + 1;
        end
      else
        if RespEsAPA[j] > RespOKDePA[i] then i := i + 1
        else j := j + 1;
      end;
    end;
```

```
begin
{
  Lee las respuestas del estudiante. Subcadenas separadas
  por comas. Las separa y ordena ascendentemente en Vector
  local Array of cad4.
  Compara con respuestas correctas en vectorglobal of cad4.
  Elimina las correctas del vector global.
  muestra correctas y pide nuevamente respuesta hasta que
todas las asociaciones se den.\
  Registra en NOTA el numero de intentos del estudiante.
}
For i := 1 to MaxAsociaciones do { iniciar vector de
                                   respuestas correctas }
  AciertosCi] := '';
NroIntentos := 0;
NroAciertos := -1;
Writeln;
Writeln;
Write('Su respuesta? ');
LineaCursorRespuesta := LinCur;
ColumnaCursorRespuesta := ColCur;
Repeat
  NroIntentos := NroIntentos + 1;
  Gotoxy(ColumnaCursorRespuesta,LineaCursorRespuesta);
  For i := 0 to NroAciertos do Write(AciertosCi],',');
  l := LinCur;
  c := ColCur;
  BUFFER255 := '';
  LeaCad(c,l,(NroAsocs+1)*5);
  AccionPA2(BUFFER255,RespEsAPA,NroAsEst,false);
  chequeo;
until NroAsocs < 0;
StoreNum(NroIntentos,'NOTA');
writeln;writeln;
end;

Procedure PPruebaAsociacion;
var CodEva: Codigo;
    Pregunta,RespuestaOK: Cad255;
    Peso: Real;
    NroAsocs: integer;
Begin
PCodEva(CodEva);
PPregunta(pregunta);
AccionPA1(pregunta);
Prespuesta(RespuestaOK);
AccionPA2(RespuestaOK,RespOKDePA,NroAsocs,true);
PPeso(peso);
VPeso(peso);
AccionPA3(NroAsocs);
Pfinal;
end;
```

```
Procedure PLimpiarPantalla;
begin
ClrScr;
Pfinal;
end;

Procedure PArchivo(Var S: Cad255);
begin
PTextoAcotado(S);
end;

Procedure AccionPI(Archivo: Cad255);
begin
close(curso);
seek(AD,NroRegEst);
read(AD,RegEst);
RegEst.Archivo := Archivo;
Seek(Ad,NroRegEst);
write(AD,RegEst);
Writeln;
Writeln('Usted ha alcanzado un punto de la leccion en el que
         puede suspender');
 Writeln('su estudio y continuarlo en otra sesion. Desea
          Continuar? (S/N) '); Repeat
  Readln(SioNo);
Until SioNo in ['s','S','n','N'];
if SioNo in ['n','N'] then
  begin
  AbandonarEnPI := True;
  end
else
  begin
  AbandonarEnPI := false;
  LeccionActual := Archivo;
  if Existe(LeccionActual) then Abrircurso(LeccionActual)
  else
    begin
    ErrorFatal('No existe leccion '+LeccionActual);
    AbandonarEnPI := true;
    end;
  end;
end;

Procedure PPuntoInterrupcion;
var Archivo: cad255;
begin
PArchivo(Archivo);
Pfinal;
AccionPI(Archivo);
end;
```

```
Procedure AccionAsignar(   NombreVariableObjeto: Cad12;
                           VariableAsignada: TVariable  );
begin
  Case VariableAsignada.TipoDeVar of
  NUMERICO: StoreNum(VariableAsignada.Valor,
                     NombreVariableObjeto);
  CARACTER: StoreCad(VariableAsignada.Cadena^,
                     NombreVariableObjeto);
  end; { case }
end;

Procedure PTermino(Var V: TVariable);
var Fac1: TVariable;
    Tk: Rtoken;
    c:  Char;
    cc: String[1];
    ValSigno: Integer;
    Bsigno: boolean;   { true si el signo no es nulo }

    Procedure PSigno;
    begin
    BSigno := true;
    c := getchar;
    while EsEspacio(c) do c:=getchar;
    case c of
      '+': ValSigno := 1;
      '-': ValSigno := -1;
      else
          begin
          ungetchar(c);
          ValSigno := 1;
          BSigno := false;
          end;
     end; {case}
   end; {PSigno}
```

```
Procedure PFactor(Var V: TVariable);
var V1: TVariable;
    Tk: Rtoken;
    c:  Char;
    cc: String[1];
    PreP,P: PTVariable;
    Estado: Integer;
begin
GetToken(Tk);
Case Tk.Codigo of
  NoLogico:
    begin
    ErrorFatal('NoLogico '+Tk.Valor);
    PFactor(V1);
    if v1.valor <> 0 then v1.valor := 0
    else v1.valor := 1;
    end;
  Variable:
    begin
    UbicarVariable(Prep,P,Tk.Valor);
    if p=nil then
      begin
      ErrorFatal('FACTOR: Variable no definida: '+Tk.Valor);
      exit;
      end;
    V := P^;
    if V.TipoDeVar = CARACTER then
      begin
      New(V.Cadena);
      V.Cadena^ := P^.Cadena^;
      end;
    end;
  CteNumerica:
    begin
    V.TipoDeVar := Numerico;
    Val(Tk.Valor,V.Valor,Estado);
    if estado <> 0 then
      begin
      errorfatal('FACTOR: Constante invalida en expresion');
      exit;
      end;
    end;
```

```
      ParentIzq:
        begin
        PExpresion(V);
        GetToken(Tk);
        if tk.codigo <> Parentder then
          begin
          ErrorFatal('Expresion invalida.
                      Parentesis desbalanceados');
          exit;
          end;
        end;
      else
        Begin
        ErrorFatal('FACTOR: Expresion invalida');
        exit;
        end;
      end; (case)
    end; { PFactor }

    Procedure OpMul;
    begin
    GetToken(Tk);
    if not (Tk.Codigo in [Multiplicar,Dividir,YLogico]) then
      begin
        Tk.Codigo := NoToken;
        ErrorFatal('Operador aritmetico invalido');
        exit;
      end;
    end; {OpMul}

    Procedure AccionTer;
    begin
{ ------------------------------ }
    If (V.TipoDeVar <> Fac1.TipoDeVar) then
      begin
      ErrorFatal('Inconsistencia de tipos entre terminos
                  para exp. simple');
      exit;
      end;
    if V.TipoDeVar <> Numerico then
      begin
      ErrorFatal('No utilizar cadenas como factores');
      exit;
      end;
      case Tk.Codigo of
      Multiplicar,YLogico:  V.Valor := V.Valor * Fac1.Valor;
      Dividir:              V.Valor := V.Valor / Fac1.Valor;
      end;
{ ------------------------------ }
    end; {AccionTer}
```

```
begin
c := getchar;
while EsEspacio(c) do c := getchar;
if c = '"' then
  begin
  { armar cadena }
  V.TipoDeVar := CARACTER;
  New(V.Cadena);
  V.Cadena^ := '';
  c := getchar;
  while c <> '"' do
    begin
    cc := ' ';
    cc[1] := c;
    V.Cadena^ := V.Cadena^ + cc;
    c := getchar;
    end; {while}
  end
 else
  begin
  UngetChar(c);
  PSigno;
  PFactor(V);
  If (V.TipoDeVar = CARACTER) then
    begin
    if (BSigno) then
      begin
      ErrorFatal('TERMINO: Expresion invalida con
                          variable Alfanumerica');
      exit;
      end
    else exit
    end
  else if V.TipoDeVar = Numerico
  then V.Valor := V.Valor * ValSigno;
  c := getchar;
  while EsEspacio(c) do c := getchar;
  while c in ['*','/','y','Y'] do
    begin
    ungetchar(c);
    OpMul;
    PFactor(Fac1);
    AccionTer;
    c := getchar;
    while EsEspacio(c) do c := getchar;
    end;
  ungetchar(c);
  end; { else }
end;   { PTermino }
```

```
Procedure PExprSimple(Var V:TVariable);
var Ter1: TVariable;
    Tk: Rtoken;
    c:  Char;
    cc: String[1];

    Procedure POpAd;
    begin
    GetToken(Tk);
    if not (Tk.Codigo in [Mas,Menos,OLogica]) then
      begin
        Tk.Codigo := NoToken;
        ErrorFatal('Operador aritmetico invalido');
        exit;
      end;
    end; {popad}

    Procedure AccionExpSim;
    begin
    If V.TipoDeVar <> Ter1.TipoDeVar then
      begin
      ErrorFatal('Inconsistencia de tipos entre terminos
                  para exp. simple');
      exit;
      end;
    if V.TipoDeVar = CARACTER then
      if Tk.Codigo = Mas then
        begin
        V.Cadena^ := V.Cadena^+Ter1.Cadena^;
        end
      else
        begin
        ErrorFatal('Expresion invalida. Operador invalido
                    entre cadenas');
        exit;
        end
    else
      begin
        case Tk.Codigo of
        Mas,OLogica:  V.Valor := V.Valor + Ter1.Valor;
        Menos:        V.Valor := V.Valor - Ter1.Valor;
        end;
      end;
    end;
```

```
begin
PTermino(V);
c := getchar;
While EsEspacio(c) do c := getchar;
while c in ['+','-','o','O'] do
  begin
  ungetchar(c);
  POpAd;
  PTermino(Ter1);
  AccionExpSim;
  c := getchar;
  While EsEspacio(c) do c := getchar;
  end;
ungetchar(c);
end;


Procedure PExpresion ( (var ValExpr: TVariable) ) ;
var ExprSimp1,ExprSimp2: TVariable;
    Tk: RToken;
    c: char;

    Procedure Relop;
    begin
    GetToken(Tk);
    if not (Tk.Codigo in [MenorQue,MayorQue,Diferente,
                          MenorOIgual,MayorOIgual,Igual]) then
      begin
        Tk.Codigo := NoToken;
        ErrorFatal('Operador Relacional invalido');
        exit;
      end
    else
      ValExpr.TipoDeVar := NUMERICO;
    end; {relop}
```

```
Procedure AccionExp;
begin
if ExprSimp1.TipoDeVar <> ExprSimp2.TipoDeVar then
   begin
   ErrorFatal('Inconsistencia en tipos de operandos
               en expresion');
   exit;
   end;
Case Tk.Codigo of
    MenorQue:
          begin
          if ExprSimp1.TipoDeVar = NUMERICO then
            if ExprSimp1.Valor < ExprSimp2.Valor then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0
          else
            if ExprSimp1.Cadena^ < ExprSimp2.Cadena^ then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0;
          end;
    MayorQue:
          begin
          if ExprSimp1.TipoDeVar = NUMERICO then
            if ExprSimp1.Valor > ExprSimp2.Valor then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0
          else
            if ExprSimp1.Cadena^ > ExprSimp2.Cadena^ then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0;
          end;
    Diferente:
          begin
          if ExprSimp1.TipoDeVar = NUMERICO then
            if ExprSimp1.Valor <> ExprSimp2.Valor then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0
          else
            if ExprSimp1.Cadena^ <> ExprSimp2.Cadena^ then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0;
          end;
```

```
    MenorOIgual:
          begin
          if ExprSimp1.TipoDeVar = NUMERICO then
            if ExprSimp1.Valor <= ExprSimp2.Valor then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0
          else
            if ExprSimp1.Cadena^ <= ExprSimp2.Cadena^ then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0;
          end;
    MayorOIgual:
          begin
          if ExprSimp1.TipoDeVar = NUMERICO then
            if ExprSimp1.Valor >= ExprSimp2.Valor then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0
          else
            if ExprSimp1.Cadena^ >= ExprSimp2.Cadena^ then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0;
          end;
    Igual:
          begin
          if ExprSimp1.TipoDeVar = NUMERICO then
            if ExprSimp1.Valor = ExprSimp2.Valor then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0
          else
            if ExprSimp1.Cadena^ = ExprSimp2.Cadena^ then
              ValExpr.Valor := 1
            else
              ValExpr.Valor := 0;
          end;
  else
    ErrorFatal('Expresion Invalida');
  end; { case }
end; {AccionExp}
```

```
begin
PExprSimple(ExprSimp1);
c := getchar;
While EsEspacio(c) do c := getchar;
if c in ['<','>','='] then
  begin
  ungetchar(c);
  Relop;
  PExprSimple(ExprSimp2);
  AccionExp;
  end
else
  begin
  if not (c in [';',')']) then
    begin
    ErrorFatal('Expresion invalida. Probablemente faltan
                parentesis o ";"');
    end;
  ungetchar(c);
  ValExpr := ExprSimp1;
  end;
end;

procedure PAsignacion(NombreVariableObjeto: Cad12);
var ValorExpresion: TVariable;
    PVar: PTVariable;
    Tk: RToken;
    c: char;
begin
GetToken(Tk);
If Tk.Codigo <> Asignacion then
  begin
  ErrorFatal('Error de Sintaxis.  -->
              '+NombreVariableObjeto+Tk.Valor);
  exit;
  end;
PExpresion(ValorExpresion);
AccionAsignar(NombreVariableObjeto, (:=) ValorExpresion);
c := getchar;
while EsEspacio(c) do c:=getchar;
if c <> ';' then
  begin
  ungetchar(c);
  errorfatal('AVISO: Falto ";" al final de expresion');
  end;
pfinal;
end;
```

```
Procedure PTextoCondicional;
var V: Tvariable;
    c: char;

  procedure AccionSi;
  var contador : integer;
      tk: RToken;
  begin
  if v.valor = 0 then
    begin                (3)
    contador := 1;
    while (contador > 0) and (not eof(curso)) do
      begin
      c := getchar;
      if c = '.' then (1)
        begin
        gettoken(tk);
          case tk.codigo of
          SI: contador := contador + 1;
          FINSI: contador := contador - 1;
          end; (case)
        end;          (1)
      pfinal;
      end; ( while )
      if contador > 0 then
        begin  (2)
        ErrorFatal('No existe balance entre "SI" y "FINSI"
                    en textos condicionales');
        exit;
        end;   (2)
    end;                  (3)
  end; (AccionSi)

begin
PExpresion(v);
c := getchar;
while EsEspacio(c) do c:=getchar;
if c <> ';' then
  begin
  ungetchar(c);
  errorfatal('AVISO: Falto ";" al final de expresion');
  end;
pfinal;
accionsi;
end;
```

```
procedure PFinTextoCondicional;
begin
pfinal;
end;

Procedure PRegistro;
Var Tk: Rtoken;
    RNota,RegEst: RegDesEst;
    P,PreP: PTVariable;
    PRDE: Integer;          { ^RegDesEst }
    Anterior: Integer;      {      "     }
    PRR,NRO_REG_RESP: Integer;          { ^Reg_Respuesta }
    RResp,Libre,Cero: Reg_Respuestas;
begin
GetToken(Tk);
Case Tk.Codigo of
N:
  begin
  UbicarVariable(PreP,P,'CODEVA');
  RNota.Cod_Eva := P^.Cadena^;
  UbicarVariable(PreP,P,'NOTA');
  RNota.Puntaje := P^.Valor;
  UbicarVariable(PreP,P,'PESO');
  RNota.Peso := P^.Valor;
  RNota.Prx_Eva := 0;
  seek(AD,NroRegEst);
  read(AD,RegEst);
  PRDE := RegEst.Pra_Eva;
  Anterior := 0;
  while PRDE <> 0 do
    begin
    seek(ad,PRDE);
    read(ad,RegEst);
    Anterior := PRDE;
    PRDE := RegEst.Prx_Eva;
    end; { while }
  if Anterior <> 0 then
    begin
    RegEst.Prx_Eva := filesize(ad);
    seek(ad,Anterior);
    write(ad,RegEst);
    seek(ad,RegEst.Prx_eva);
    end
  else
    begin
    RegEst.Pra_Eva := filesize(ad);
    seek(ad,NroRegEst);
    write(ad,RegEst);
    seek(ad,RegEst.Pra_Eva);
    end;
  write(ad,RNota);
  end; { N: }
```

```
R:
  begin
  if not existe(ArchRespuestas) then
    begin
    assign(ar,archrespuestas);
    rewrite(ar);
    cero.prim_eval := 0;
    cero.Prim_libre := 0;
    write(ar,cero);
    close(ar);
    end;
  assign(ar,ArchRespuestas);
  reset(ar);
  seek(ad,NroRegEst);
  read(ad,RegEst);
  RResp.Nro_Est := RegEst.Nro_Est;
  UbicarVariable(PreP,P,'CODEVA');
  RResp.Cod_Eva := P^.Cadena^;
  UbicarVariable(PreP,P,'PESO');
  RResp.Peso_eva := P^.Valor;
  UbicarVariable(Prep,P,'RESPUESTA');
  RResp.respuesta := P^.Cadena^;
  RResp.proximo := 0;
  seek(ar,0);
  read(ar,cero);
  if cero.prim_libre <> 0 then
    begin
    NRO_REG_RESP := Cero.prim_libre;
    seek(ar,NRO_REG_RESP);
    read(ar,libre);
    cero.prim_libre := libre.proximo;
    end
  else
    begin
    NRO_REG_RESP := filesize(ar);
    end;
  PRR := cero.prim_eval;
  anterior := 0;
  while PRR <> 0 do
    begin
    seek(ar,PRR);
    read(ar,libre);
    Anterior := PRR;
    PRR := libre.proximo;
    end; { while }
  if anterior = 0 then
    BEGIN
    cero.prim_eval := NRO_REG_RESP;
    END
```

```
  else
    begin
    libre.proximo := NRO_REG_RESP;
    seek(ar,anterior);
    write(ar,libre);
    end;
  seek(ar,NRO_REG_RESP);
  write(ar,RResp);
  seek(ar,0);
  write(ar,cero);
  close(ar);
  end; { R: }
else
  ErrorFatal('ERROR: RG ? - La orden de registro no trae
                           el parametro apropiado'); end;
{case} Pfinal; end;

procedure pcomando;
var c: char;
    i: integer;
    Tk: RToken;
begin
i:= 0;
GetToken(Tk);
case Tk.Codigo of
Et:PEspereTecla;
Pc:PPruebaCorreccion;
Fi:PFraseIncompleta;
Rc:PRespuestaConstruida;
Pa:PPruebaAsociacion;
Pito:PSonido;
Lp:PLimpiarPantalla;
Si:PTextoCondicional;
FinSi:PFinTextoCondicional;
Pi:PPuntoInterrupcion;
Co:PComentario;
Rg:PRegistro;
Variable:PAsignacion(Tk.Valor);
else ErrorFatal('Comando inexistente');
end; { case }
end;
```

```
procedure presentar;
var c: char;
    i: integer;
    TV: TVariable;
begin
i := 2;
while (not EOF(curso)) and (not AbandonarEnPI) do
  begin
  c := getchar;
  if c in [#10,#13] then
    begin
    i := i + 1;
    if i>2 then i:= i-2;
    write(c);
    end
  else if (c = ',') and (i = 2) then pcomando
  else if (c = ' ') then
    begin
    PExpresion(TV);
        Case TV.TipoDeVar of
        NUMERICO: Write(TV.Valor:15);
        CARACTER: Write(TV.Cadena^);
        end; { case }
        c := getchar;     { Desecha ";" final de la expresion }
    end
  else
    begin
    write(c);
    i:=0;
    end;
  end; { while }
  If not AbandonarEnPI then
    begin
    seek(AD,NroRegEst);
    read(AD,RegEst);
    RegEst.Nro_Lec := RegEst.Nro_Lec + 1;
    RegEst.Archivo := RL[RegEst.Nro_Lec].Nom_Lec;
    Seek(Ad,NroRegEst);
    write(AD,RegEst);
    Writeln;
    Write('Fin de leccion. Presionar ENTER');
    AccionET(#13);
    end;
end; {presentar}
```

```
Function ObtenerLeccion: Boolean;
Var Resultado,Final : Boolean;

  Procedure ValidarAcceso;
  var CodEst: Codigo;
  begin
  ClrScr;
  Write('Numero de estudiante: ');
  BUFFER255:='';
  LeaCad(22,1,6);
  CodEst := BUFFER255;
  NroRegEst := BusCod(CodEst);
  end;

  Procedure AsignarLeccion;
  begin
  seek(ad,NroRegEst);
  read(ad,RegEst);
  If RegEst.Archivo = '' then
    begin
    RegEst.Archivo := RL[RegEst.Nro_Lec].Nom_Lec;
    seek(ad,NroRegEst);
    write(ad,RegEst);
    end;
  LeccionActual := RegEst.Archivo;
  If LeccionActual = '' then
    begin
    Resultado := false;
    Final := true;
    end
  else Resultado := true;
  end;
```

```
 begin
 Final:= false;
 ValidarAcceso;
 Resultado := (NroRegEst > 0);
 If Resultado then AsignarLeccion
 else ErrorFatal('No está registrado en este curso');
 If Resultado then
   if not existe(LeccionActual) then
     begin
     ErrorFatal('No existe la leccion '+LeccionActual);
     Resultado := False;
     end;
 If final then
   begin
   ClrScr;
   Writeln;
   write('Fin del curso.');
   end;
 ObtenerLeccion := Resultado;
 end;

 function iniciar: boolean;
 var i: integer;
 begin
 LastToken.Codigo := NoToken;
 LastToken.Valor := '';
 nextunget := 1;
 nextget   := 1;
 for i:=1 to MAX_UNGET do lastchar[i] := char(0);
 if (not existe(ArchDesempeno)) then
   begin
   errorfatal('No existe archivo de estudiantes (desempeño)');
   iniciar := false;
   exit;
   end;
 if (not existe(ArchLecciones)) then
   begin
   ErrorFatal('No existe archivo de lecciones');
   iniciar := false;
   exit;
   end;
 assign(ad,archdesempeno);
 reset(ad);
 assign(al,ArchLecciones);
 reset(al);
```

```
 for i := 0 to Max_Lec do
   BEGIN
   read(al,rl[i]);
   END;
 if (not existe(ArchRespuestas)) then
   begin
   assign(AR,ArchRespuestas);
   rewrite(AR);
   RR.prim_eval := 0;
   RR.prim_libre := 0;
   write(AR,RR);
   close(AR);
   end;
 assign(AR,ArchRespuestas);
 reset(AR);
 primvar := nil;
 DefinirVariable(Numerico,'NOTA');
 DefinirVariable(Caracter,'RESPUESTA');
 DefinirVariable(Numerico,'PESO');
 DefinirVariable(Caracter,'CODEVA');
 IniciarTablaToken;
 AbandonarEnPI := false;
 iniciar := true;
 end;

 procedure ensayos;
 var PreP,P:PTVariable;
 begin
 {
   UBICARVARIABLE(PREP,P,'RESPUESTA');
   WRITELN('RESPUESTA = ',P^.CADENA^);
 }
 P:= PrimVar;
 While p <> NIL do
   begin
   writeln(p^.NombreVar);
     case p^.tipodevar of
     numerico: Writeln('Numerica - ',P^.Valor);
     Caracter: Writeln('Caracter - ',P^.Cadena^);
     end; {case}
   readln;
   p := p^.proxvar;
   end;
 end;
```

```
begin (main)
if not iniciar then exit;
While ObtenerLeccion do
  begin
  abrircurso(LeccionActual ('toro0.'));
  presentar;
{  ensayos; }
  if not AbandonarEnPI then
    begin
    close(curso);
    ClrScr;
    WriteLn('Desea iniciar la siguiente lección? (S/N)');
    Repeat
      Readln(SioNo);
    Until SioNo in ['s','S','n','N'];
  end;
  if (SioNo in ['n','N']) or (AbandonarEnPI) then
    begin
    close(ad);
    close(al);
    close(ar);
    clrscr;
    exit;
    end;
  end;
end. (main)
```